(1)

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/88D-22

DATABASE DESIGN AND LAND BATTLE INTERFACE FOR THE FAST STICK EXERCISE

THESIS

Swen Walker
Captain, USAF

AFIT/GCS/ENG/88D-22

AFIT/GCS/ENG/88D-22

DATABASE DESIGN AND LAND BATTLE INTERFACE FOR THE FAST STICK EXERCISE

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science (Computer Science)

Swen Walker, B.S.

Captain, USAF

December, 1988

Preface

The goal of this thesis was to enhance the FAST STICK exercise by addressing some of its current limitations. To accomplish these enhancements, a relational database system was designed and implemented to store FAST STICK data, an interface between FAST STICK and JPLAN was developed, and the exercise simulation was extended to include simulation of land play. The FAST STICK exercise is a tactical air employment wargame that presents the different tasks of air power associated with air combat operations, the interaction of these tasks, and their application in a joint combat operational environment.

This thesis presents background on the FAST STICK exercise, limitations of the current simulation, and solutions to those limitations through the use of a database management system and an extension of the simulation.

I am deeply indebted to my thesis advisor, Captain Mark Roth, for his invaluable assistance during the development of this thesis. I also wish to thank the other members of my committee, Lt. Colonel Skip Valusek and Major Dan Reyen, for their assistance. I would also like to thank Captain Ken Wilcox, for his endless hours of help with the INGRES fourth generation language. Finally, I would like to thank my wife, [redacted] for her unwavering support, editing assistance, and understanding throughout this project.

Swen Walker

Table of Contents

List of Figures

List of Tables

Abstract

FAST STICK is an interactive, conventional, tactical air employment war simulation. The model is a teaching vehicle which allows users to apply basic, tactical employment concepts of air superiority, interdiction, close air support, and reconnaissance. The primary user of this model is the Air Command and Staff College in their theater warfare curriculum. The current version of FAST STICK was converted from Honeywell mainframe Fortran to IBM compatible, microcomputer Pascal by programmer/analysts from the Air Force Wargaming Center in 1987. Although major improvements were made to the program, the exercise still had shortcomings and limitations as a joint exercise. Specifically, limitations were identified in the following areas: a) the simulation exercises' use of flat files to store data; b) the lack of an automated data interface between FAST STICK and its deployment planning counterpart, JPLAN; c) the lack of land simulation play in this joint exercise.

The goal of this thesis effort was to enhance the FAST STICK exercise by addressing these limitations. This was accomplished by incorporating a relational database management system (INGRES) into the exercise to allow game controllers to better manage the FAST STICK data. Using the system level facilities and the power of the SQL language found in the INGRES database management system, an automated interface was constructed between the new FAST STICK database and the existing JPLAN database. Finally, an extension to the FAST STICK simulation was developed that permitted game controllers to incorporate different land battle scenarios into the exercise.

Overall, the environment the FAST STICK game controllers operate in to run the simulation program has improved. By enhancing the controllers' access to the exercise data through the use of a database management system, automating the transfer of data between JPLAN and FAST STICK, and adding land battle events to the simulation, the exercise can now be easily modified or tuned by game controllers to meet changing learning objectives and doctrine in the area of air combat operations in a joint military operational environment.

DATABASE DESIGN AND LAND BATTLE INTERFACE FOR THE FAST STICK EXERCISE

I. Introduction

1.1 Background

FAST STICK is a computer simulated tactical air employment exercise which serves as the capstone for the theater warfare phase of the Air Force's Air Command and Staff College curriculum [2:1-1]. Its main objective is to provide intermediate level Air Force staff officers the opportunity to apply the basic tactical employment concepts of reconnaissance, counter air, interdiction, and close air support. The game is the final step in a joint planning exercise to deploy and employ air forces against the forces and targets of an imaginary enemy.

Deployment planning of forces is done with another simulation program known as JPLAN (Joint Planning Exercise) while employment planning and execution of forces is done in FAST STICK. The output of JPLAN, which is combat forces information, is used as input to the FAST STICK exercise.

The general scenario of the FAST STICK exercise depicts two fictitious countries (Brazona and Iguana) on the brink of open hostilities. Following Brazonan attacks on Iguanan forces, the Iguanan government has requested the introduction of U.S. forces to help defend Iguana. The U.S. National Command Authority has authorized the deployment of a military joint task force to defend U.S. citizens in Iguana and Iguana from Brazonan aggression. The Air Force component of this task force is responsible for conducting an aggressive 72-hour counter air campaign to effectively neutralize the Brazonan air threat It is against this backdrop that FAST STICK occurs. The FAST STICK game attempts to simulate the environment that an individual would be exposed to in the plans and operations branches of a Tactical Air Control Center (TACC) during this 72 hour tactical campaign.

In the exercise, individuals are members of a team that make up the TACC branches. Each team member performs a staff function of one of these branches. During the exercise, team members determine the priority of targets to be destroyed, assign a desired damage expectancy for each target, plan reconnaissance missions to obtain more information, and then decide on which targets to attack. Each team starts with a limited number of aircraft resources and is expected to meet stated objectives from higher command directives. The FAST STICK simulation program accepts a team's flight plans as input, simulates the missions, and returns the results at the end of each game cycle to allow them to make any changes to their next cycles' flight plans. At the end of the exercise, the program compiles a score based on targets hit and resources remaining. This score along with summary reports of missions flown during the exercise are used to review and critique a team's performance.

FAST STICK was originally written in Fortran and ran on a Honeywell H6000 mainframe computer. Input and output to the program was accomplished over a 300 baud hard copy terminal device. The interface was very user unfriendly, as input into the program had to follow a rigid format. Players would spend more time learning the computer syntax required to input data than playing the game. The program was also very inflexible. Changes could not be easily made to the game's scenario, parameters, or data without a major effort each time.

In August of 1987, the staff of Air Force Wargaming Center began a rapid prototyping effort to improve FAST STICK by rehosting it on a Zenith 158 microcomputer and modifying the user interface. The new simulation program was written in Pascal, and the user interface was replaced by a screen-oriented menu driven system. Although major improvements were made to the program, the exercise still had shortcomings and limitations as a joint planning exercise.

1.2 Problem Statement

The FAST STICK simulation program had several shortcomings. They were:

1. Exercise data was stored in a flat file storage structure and encoded with special numeric coding formats. This method of storage made it very difficult for game controllers to change the data or parameters of the game in order to calibrate the exercise for different scenarios or learning objectives. The flat file structure and data encoding also resulted in much data redundancy with no means of ensuring data integrity and consistency.

2. There was no automated data interface to transfer data between the JPLAN exercise and the FAST STICK exercise. JPLAN is used earlier in the theater warfare phase of the curriculum to identify combat forces, and build force lists to be used in the FAST STICK campaign. However, all data generated by JPLAN had to be manually transferred between the two systems by the game controller.

3. There was very little land simulation play in the game. The program was supposed to simulate a joint exercise, but the only land simulation was occasional random requests for close air support. These requests were usually compietely unrelated to each other, and had no relevance to any type of concerted effort on the part of either side in the conflict to achieve some tactical land or battle objective.

1.3 Scope

The purpose of this thesis effort was to enhance the FAST STICK exercise by addressing the shortcomings and limitations mentioned in the problems section above. The FAST STICK exercise was enhanced by designing and implementing a database system to manage and manipulate the exercise data. Additionally, an automated interface was developed between the JPLAN and FAST STICK exercise to permit transfer of combat forces data. Finally, a scenario generation program was developed to add land simulation play to the exercise.

1.4 Justification

As noted in Smith, operationally oriented exercises such as FAST STICK are the best learning tools Air University has to teach force requirements and general force employment

[17:683]. Such exercises can be constrained by the hardware and software technological capabilities available at the time of their development. These limiting capabilities have resulted in significant limitations to the computerized wargaming activities at Air University [17:683]. An example of such limitations in FAST STICK was the inflexibility of the exercise to permit easy modification of forces and scenario data to reflect different tactical environments or emphasize different concepts. Additionally, the lack of any type of land play made it hard to represent real world scenarios.

The FAST STICK program used flat files to store the multitude of probabilities and parameters needed to simulate the various aircraft missions. Because this data was stored in flat files, changes to the data by game controllers in order to tune the exercise for different scenarios or learning objectives was very difficult. The data for the most part could only be changed one item at a time. To do changes on several data items in the files required a great deal of time under the storage scheme because each item had to be individually changed one at a time. The program also used numeric codes to represent all of the data in the flat files. For example, the aircraft type F4E is represented by the integer 1. The use of names to represent the various parameters would have been much more meaningful.

In order to alleviate the problems of the flat file structure, the game controllers needed a tool that would allow them to easily view or update the exercise data. A commercial database management system with a fourth generation application development tool would provide the flexibility to manipulate the multitude of data the exercise uses. The DBMS would also provide data integrity and consistency, eliminate data redundancy, and provide a user interface that would permit controllers to easily modify data and generate reports on exercise results.

As stated earlier, one of the objectives of the FAST STICK exercise is to show the interaction between tasks of air power and their application in a joint combat operation [3:4]. However, in the area of the application of air support to ground actions, the exercise is limited. Although there are up to four requests for close air support per day during the exercise, the player is at a disadvantage because he is unable to perceive actual land battles taking place or the results of his air support. By adding land battle simulation to

the exercise, the player will be able to more fully comprehend how his actions influence the outcome of ground actions.

1.5 Assumptions

The enhancement of the FAST STICK simulation program was based on the following assumptions:

1. The enhanced version of the FAST STICK program would be considered valid if it produced output identical to the original implementation given the same input data except for the results of close air support missions.
2. The addition of land play to the simulation exercise would only affect the overall play of the exercise by generating aircraft requests for close air support, loss of aircraft on close air support missions, and the presentation of land battle events to players. The land battle simulation would execute preset scenarios. The outcome of these scenarios would be affected by the number of aircraft a player had allocated to close air support in the exercise.
3. Since the addition of land play to the exercise was to emphasize the importance that close air support plays in land battles rather than players becoming familiar with ground combat, players would not input any type of land battle data.

1.6 Approach/Methodology

Solutions to the shortcomings of the FAST STICK exercise were accomplished in a three phase effort. In the first phase, the database design and implementation were developed in order to solve the problems encountered because of the flat file structure. Within the second phase, an automated interface between the JPLAN and FAST STICK exercise was developed using facilities provided by the database management system. In the third phase, a land battle event generation program was developed that added land battle scenarios to the exercise. This program was integrated into the aircraft mission simulation routines.

The following list outlines the steps that were followed in accomplishing this thesis project:

1. Database Design and Implementation

   (a) An entity-relationship(E-R) diagram was developed by analyzing the FAST STICK program and flat file structure, examining the input and output requirements of the exercise, and reviewing user and maintenance manuals. An entity- relationship diagram is a graphical representation of the logical structure of a database.

   (b) The E-R diagram was implemented on a commercial database management system using database design techniques described in Chapter 3. The INGRES relational database system was selected to fulfill the FAST STICK DBMS requirements. The primary reason for the selection of this particular DBMS was that the Air Force Wargaming Center currently uses INGRES for other in-house database applications, notably JPLAN, and has the trained personnel to support and maintain an INGRES enhanced version of the exercise.

   (c) Programs were developed to upload the large quantity of exercise data from the FAST STICK flat files to the new database.

   (d) An application program was developed to allow game controllers to selectively down-load exercise data from the database to the FAST STICK flat files.

   (e) The database design, design methodology, and implementation was documented. This material was later incorporated into the FAST STICK user and maintenance manuals.

2. Automated Interface Development

   (a) An automated data interface between JPLAN and the new FAST STICK exercise was developed. JPLAN's data was stored in an INGRES database. Using the system level facilities and the power of the SQL language found in INGRES, a batch file was created containing canned query and database management

system level commands to transfer the combat forces data from the JPLAN database to the FAST STICK database.

3. Land Battle Simulation Software Addition

   (a) The FAST STICK simulation routines were analyzed to determine how the exercise simulated missions.

   (b) An event driven land battle scenario generator model was designed to incorporate land battle events into the existing simulation.

   (c) The land battle scenario generator model was implemented as a separate independent program.

   (d) The program was then integrated into the aircraft mission simulation routines.

   (e) The land simulation scenario model, data requirements, and implementation were documented. This material was later incorporated into the FAST STICK user and maintenance manuals.

1.7 Materials and Equipment

The Air Force Wargaming Center provided two Zenith-158 microcomputers with 640K RAM, color graphics adapter, color monitor, and Bernoulli technology 10Mb hard drive to run the FAST STICK exercise along with the source code and documentation for the FAST STICK program. To provide a better software development environment and accelerated compiler compilation, a Zenith 248 with 640K RAM, 80287 math coprocessor, EGA graphics capability, and a 40Mb hard drive was also made available by the Wargaming Center. They also provided the commercial database management system and associated software required to complete this project.

1.8 Thesis Outline

The remainder of this thesis is organized into four additional chapters. Chapter II describes the methodology used to develop a conceptual view of the database and the final conceptual design of the database. Chapter III presents relational database design is-

sues, the implementation of the conceptual design with a relational database management system, and describes how the automated data interface between the FAST STICK and JPLAN exercises was developed. Chapter IV briefly reviews existing literature on simulation programming of ground combat models, presents how the FAST STICK exercise simulates aircraft missions, and describes the land battle scenario generation program developed to introduce battle events to the exercise. Chapter V discusses overall conclusions and recommendations for further work.

II. FAST STICK Database Design

2.1 Database Management Systems and Design

Prior to the introduction of database management systems, most automated information processing within an organization was done through file processing. File processing involved data being stored in different record formats in files. To manipulate the data, application programs were written to extract and add records to the files. This approach to managing data had several disadvantages. Primarily, large amounts of data was duplicated throughout the files, increasing both the potential of data inconsistency and the need for additional storage capacity. Additionally, any modification to the structure of the data in the files often required changes to accompanying application programs requiring hours of unproductive maintenance programming.

With the advent of database management systems, many of the problems associated with file processing systems were eliminated. One of the reasons for this is that database systems employ the concept of data abstraction [15:4]. Data abstraction provides users "with an abstract view of the data" [15:4]. Implementation details as to how the data is stored, manipulated, or retrieved are hidden from the user. Data abstraction results in data independence. Data independence means that application programmers no longer need to be concerned about the physical schema of stored data. Consequently, application programs need not be changed or rewritten each time the physical structure of the data they operate on changes.

Although database management systems can eliminate many of the problems found in a file processing system, to be successful, they must be skillfully designed and properly implemented. In the past, database design activities have consisted of trial and error approaches using ad hoc techniques [19:3]. These types of database design often lead to inflexible solutions. A database that is poorly designed offers few advantages over a file system. In order to avoid this problem, database design must be examined from a systematic approach. In this thesis effort, two systematic approaches for the design of databases for relational systems (although one of the approaches could be applied to the design of a hierarchical or network database) were examined. Both of these approaches

involve design at the conceptual level. This is done in order to concentrate exclusively on the properties of the data so that a designer need not be concerned with the peculiarities of a particular DBMS [13:30].

Database design approaches for relational systems were examined because of the Air Force Wargaming Center's requirement that the FAST STICK database be implemented on the INGRES Relational Database Management System. As stated in chapter I, the primary reason for the selection of this particular DBMS was that the Air Force Wargaming Center currently uses INGRES for other in-house database applications, and has the trained personnel to support and maintain an INGRES enhanced version of the exercise. Additionally, there was no cost associated with employing this database system since it was purchased for use in previous thesis projects.

2.2 Database Design Methodologies

In designing the database structure for the FAST STICK exercise two database design methodologies were examined, bottom-up and top-down design approaches. In both approaches, limiting data redundancy and data inconsistency are the prime objectives. For this thesis effort, the top down approach was chosen as the design methodology because of drawbacks in the application of the bottom up approach to the FAST STICK exercise.

In the bottom-up approach, the focus is on the interdependence of data attributes [13:35]. As the title of the approach indicates, design begins by identifying the most elementary pieces of information within a database (data attributes) and then combining these units to build the tables of the database. The elementary units are characteristics of objects found within the enterprise the database is being designed for. Each attribute is associated with a column of a table in the database. The design methodology consists of the following sequence of steps [13:35]:

1. Identify all the required data attributes of interest to the enterprise.
2. Combine these attributes into tables using a normalization process to remove redundant data.

Although the bottom-up approach produces good database designs, it suffers when applied to larger database applications. Howe points out several deficiencies. In instances where an application has a large number of data attributes, the designer can become overwhelmed by the sheer number of them. Additionally, when initially beginning the database design process, the designer may not be aware of all the attributes that will be included in the design but instead that there will be a requirement for attributes about a certain object. The approach also has drawbacks in the case where there is more than one relationship between attributes. It may not always be readily clear or evident to the designer that there might be more than one relationship between attributes.

In the top down approach, a data model called the Entity-Relationship Model is used to develop an overall logical structure for a database. Developed by Chen, the model illustrates data and the relationships between data. It was conceived to facilitate database design by allowing the specification of an enterprise schema [15:21] [18:175]. "An enterprise schema represents the entire enterprises view of the data and is independent of storage or efficiency considerations" [18:175]. It represents the logical structure of the database.

This modelling technique represents the real world as a collection of objects called entities, and the relationships between these entities. An entity is an object which the enterprise recognizes as being distinguishable from other objects [15:21]. In the FAST STICK exercise, aircraft and bases are examples of entities. A relationship is an association between two or more entities such as aircraft being based at a particular airbase. Entities consist of attributes, which are properties or characteristics of the entity. For the entity *Aircraft*, two of its attributes might be *aircraft-tail-nbr* and *operational-status*.

Within the model, constraints on relationships are described in terms of mapping cardinalities. Mapping cardinalities express the number of entities to which another entity can be associated by means of a relationship [15:25]. An example of such a relationship constraint in the FAST STICK exercise is that an aircraft can only be based at one airbase, while an airbase can have several aircraft assigned to it. This is a case where a one-to-many relationship exists from the entity *Aircraft* to the entity *Airbase*. Besides the one-to-many relationship mentioned above, there are also one-to-one, many-to-one, and many-to-

many mapping cardinalities. These mapping cardinalities allow the designer to model real world constraints which the database might need to conform to in some situations.

Although, conceptually, entities are considered distinct in the model, from a database implementation standpoint, a means of identifying one entity from another entity within an entity set must be established. An entity set is a set of entities of the same type [15:21]. Such a distinction is made through the use of the entity's attributes. Within each entity set, an attribute is identified as a primary key for that entity set [15:27]. "The primary key is a set of one or more attributes, which, taken collectively, allow us to identify uniquely an entity in the entity set and a relationship in a relationship set" [15:42]. However, the possibility exists that an entity set might not have sufficient unique attributes to form a primary key. Such an entity is called a weak entity, while entities that have a primary key are called strong entities [15:29]. A weak entity cannot exist on its own; rather, it must depend on a strong entity for its existence. In order to distinguish weak entities in an entity set, the primary key from a strong entity is combined with a set of attributes from the weak entity that make it distinct [15:29].

The concepts of specialization and generalization were added to the E-R model to support relationships among closely related entity sets [15:37]. Generalization emphasizes the similarities among lower level entity sets by combining them to form higher level entity sets. Specialization emphasizes the distinction between higher level and lower level entity sets by constructing lower level entity sets from higher level entity sets [15:38]. These concepts permit designers to model the sharing of common attributes and relationships between entities that are similar.

The components and concepts discussed above can all be represented graphically in the E-R model by an E-R diagram. The E-R diagram consists of the following components:

1. Rectangles – represent entity sets.
2. Ellipses – represent attributes.
3. Diamonds – represent relationships between entity sets.
4. Lines – represent links between entity sets and relationships and attributes to entity sets.

5. Label N and Label 1 – represent the cardinality of relationships between entities.
6. Triangles – represent an ISA relationship, which is a specialization of a higher level entity or generalization of a lower level entity.

Figure 1 illustrates a small sample E-R diagram from the E-R diagram for the FAST STICK database. It represents four entities, one weak entity, four relationships, and an "ISA" specialization found in the FAST STICK database. The *Aircraft* entity represents the specific aircraft that will be used to fly the missions during the simulation. The *Airbase* entity represents the airbases where friendly aircraft are located. The *Weather Conditions* weak entity represents the times an airbase is closed due to poor weather. The *Aircraft Type* entity represents the different types of aircraft flown in the simulation. The *Mission* entity represents the different missions flown in a given cycle of the exercise. A specialization of the *Mission* entity has been included because missions can be of different types each with their own unique attributes. The *Based-At* relationship represents where each aircraft is based. The relationship is one-to-many because an aircraft can only be based at one base in any particular instance while an airbase has many aircraft assigned to it. The *Takes Off AT* relationship represents where each mission begins. Each of the entities and relationships do have attributes although they are not shown in the figure because of the tool used to draw the diagram. Further discussion of this tool will follow later in this chapter.

The sequence of operations for top down design are as follows [13:94]:

1. Identify the entities for the enterprise for which the database is being designed.
2. Identify the attributes of each entity.
3. Identify the relationships between entities and sketch the E-R diagram.

The top down design approach was selected as the database design methodology for this thesis effort for the following reasons:

1. The abstraction provided by the E–R Model permits the designer to better deal with applications that have large amounts of data.

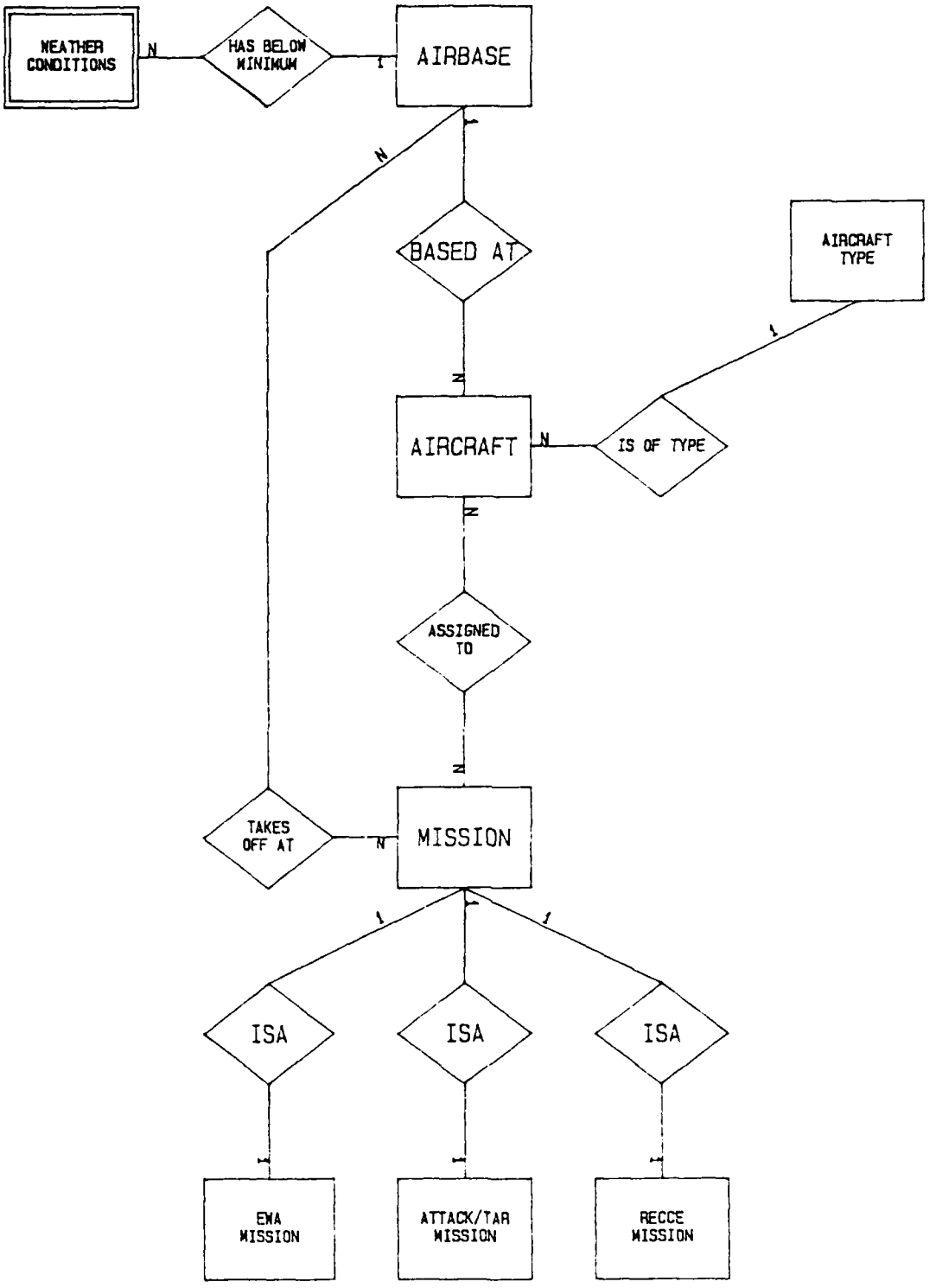

Figure 1. Sample E-R Diagram

2. The graphical presentation of the data and the relationships between the data in the E-R model easily communicates the organization of data in the design.

3. The resulting designs lend themselves better to modification because of the fact that they are not application specific.

4. The designer need not be concerned with implementation specific details for any particular database system.

5. The design approach allows designers to more easily detect and eliminate redundant attributes and relationships.

6. The E-R model with its various extensions has gained acceptance as an appropriate data model for database design and is widely used in practice [15:6] [18:188].

2.3 FAST STICK Database Design

As stated above, the top down design approach was selected to design a conceptual view of the FAST STICK database. However, rather than using pencil and paper to draw the E-R diagram, an automated structured diagrammatic tool called ER-Designer was used to generate the diagram. The tool graphically displays the diagram on a CRT screen permitting the addition or deletion of entities and relationships to the diagram. It stores all information necessary for converting the E-R diagram to a specific DBMS implementation format in an internal dictionary. Additionally, the designer tool supports the use of a plotter for drawing E-R diagrams.

The FAST STICK file structure, exercise input requirements and output results, and maintenance and operational manuals were used as a guide in designing the E-R diagram. Initially, the seventeen FAST STICK data files were analyzed to construct a preliminary diagram. The diagram was then modified several times as additional understanding and knowledge of the exercise was acquired from playing the exercise and reviewing the exercise documentation. The modifications were primarily reductions in redundant data.

The first step of the design process was identifying the entities (objects) of the enterprise (FAST STICK exercise). The following entities were identified:

- Aircraft
- Aircraft Type
- Weather Type
- Airbase
- Air Refueling Area
- Recce Target Type
- Missions
- Recce Strips
- Fixed Probabilities
- Bombs
- Pass Number
- Game Totals
- Attack Targets
- Target Type
- Flight Information
- Target Region
- EW Track
- TAR Requests
- Weapon Type
- Weapon Load
- TAR Weapon Load
- Teams

An "ISA" relationship was applied to the Mission entity for specialization. The following entities were identified:

- Electronic Warfare Mission (EWA Mission)
- Reconnaissance Mission (Recce Mission)
- Attack/Tactical Air Request Mission (Attk/Tar Mission)

An "ISA" relationship was applied to the Attack Target entity for specialization. The following entities were identified:

- AAA Site
- SAM Site
- Air Defense Radar Site
- Runway
- Interceptor AC Site
- Counter Attack AC Site

Step two in the top down approach was identifying the attributes for each of the entities. Table 1 is an example of an entity found in the FAST STICK exercise. The table contains attributes for the entity followed by a brief explanation. The format next to each of the attributes in the table is as follows: I – integer, C – character, L – logical and F – floating point. The number preceding these symbols is the length of the field. The primary keys for these entities are identified by the symbol "*" next to the attribute. A complete listing of all the entities and their associated attributes can be found in the tables in Appendix A.

The last step in the design process is to identify the relationships between the various entities and draw the E-R diagram. The final E-R diagram for the FAST STICK database

ATTRIBUTES	*FORMAT*
Aircraft Tail Number	I-3
Operational Status	C-13
Mission of Aircraft	C-13
Aircraft Out of Gas	I-1
Time Aircraft Destroyed	I-4

*

Table 1. Aircraft Entity

is shown in Figures 2, 3, 4, and 5. It consists of over 35 entities and over 60 relationships. Normally, the attributes associated with each entity and relationship are drawn as ellipses on the diagram. However, in the diagram the sheer number of the attributes clutter up the diagram so much that a lot of the understanding of the organization of the data is lost. Furthermore, the E-R Designer tool does not display attributes on the diagram; rather, it displays attribute information on a separate screen whenever attribute information is requested on an entity or relationship. The tool also does not display "ISA" specialization components as triangles; but rather, represents them as diamonds with the word "ISA" displayed inside the diamond. A listing of each the relationships in the diagram and associated attributes can also be found in the tables in Appendix A.

The *Aircraft* entity in Table 1 represents the specific aircraft that will be used to fly the missions during the simulation. The *Aircraft* entity is only used by the simulation program, not by the teams playing FAST STICK. Teams only input flight plans. The simulation program executes the plans using these aircraft.

2.4 E-R Designer Tool

As stated above, an automated designer tool was used to draw the E–R diagram. The tool had both advantages and disadvantages. One advantage was the fact that because the diagram was stored electronically, modifications to it were fairly easy to make. The tool ensured that the proper connections between the various components of the E–R diagram were correct. One limitation of the tool was that the diagram was displayed as a grid on the screen with each component placed in a cell of the grid. In some situations, where many connections were made to one component, it became very difficult to show connections without lines overlapping components. Overall, the tool was very helpful in making the

numerous revisions to the E–R diagram necessary to produce the final product. However, plotting out large diagrams with the tool required many hours of experimenting with row and column dimensions in order to arrive at the scale that provided the best representation of the diagram.

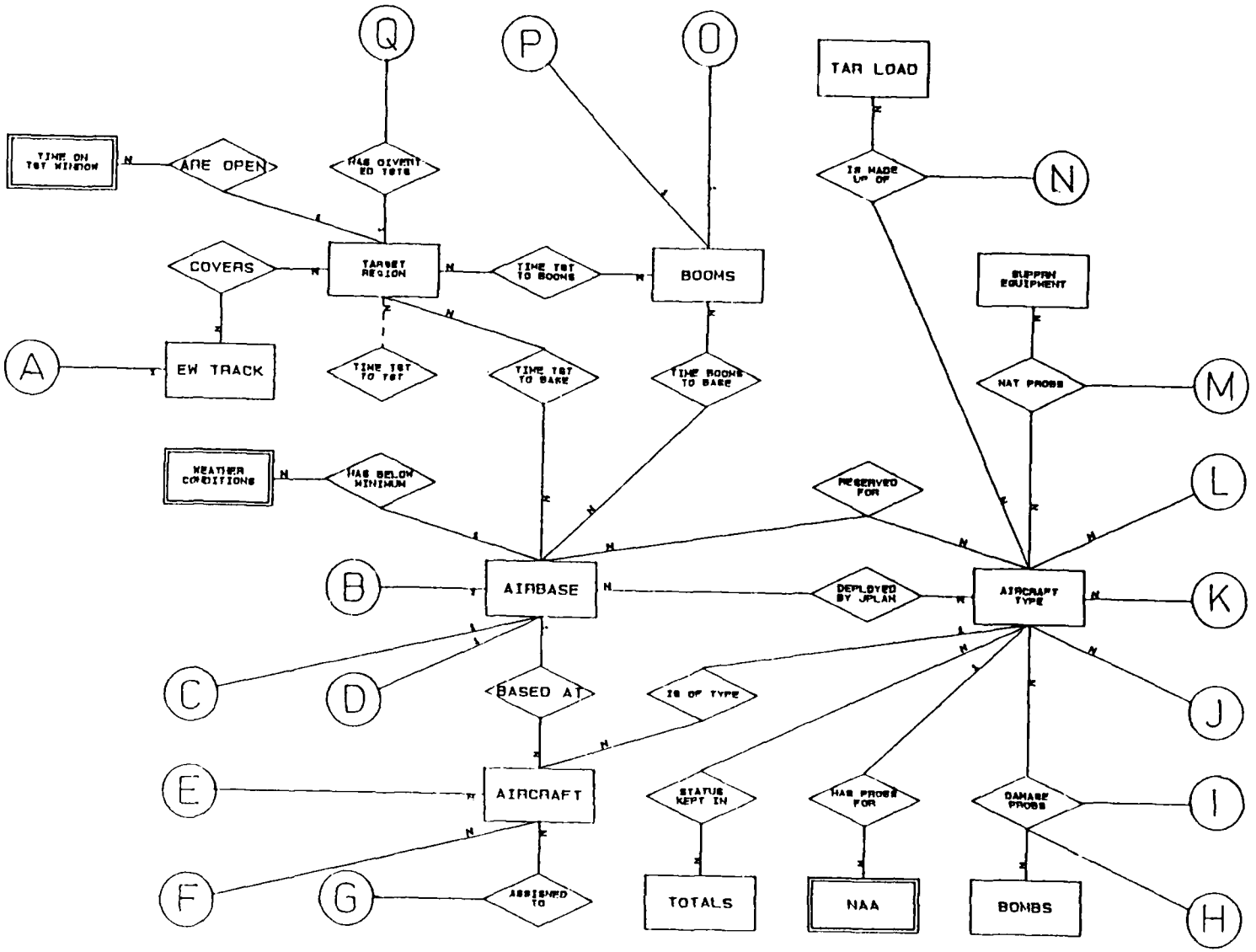

Figure 2. FAST STICK Database E-R Diagram (Part 1)

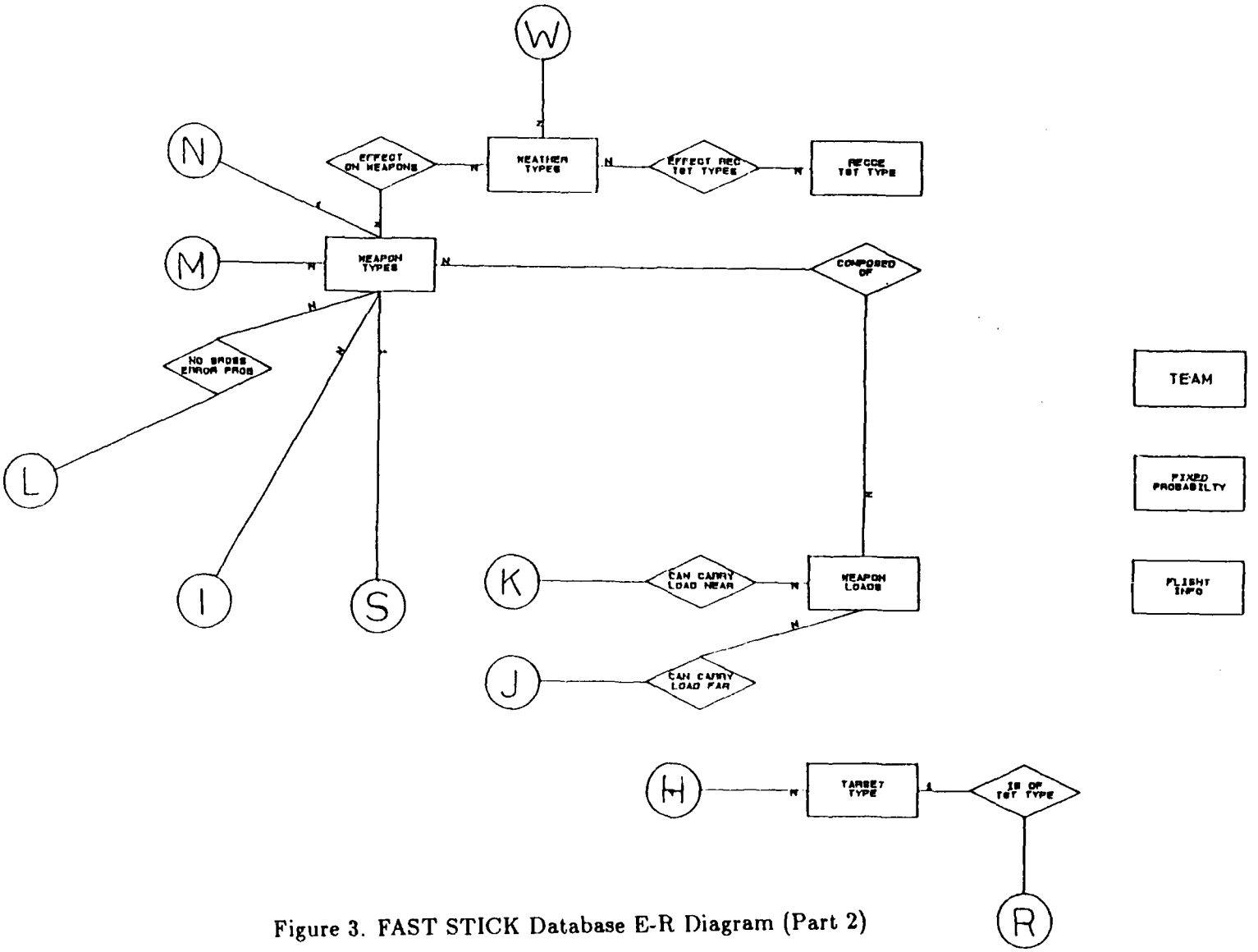

Figure 3. FAST STICK Database E-R Diagram (Part 2)

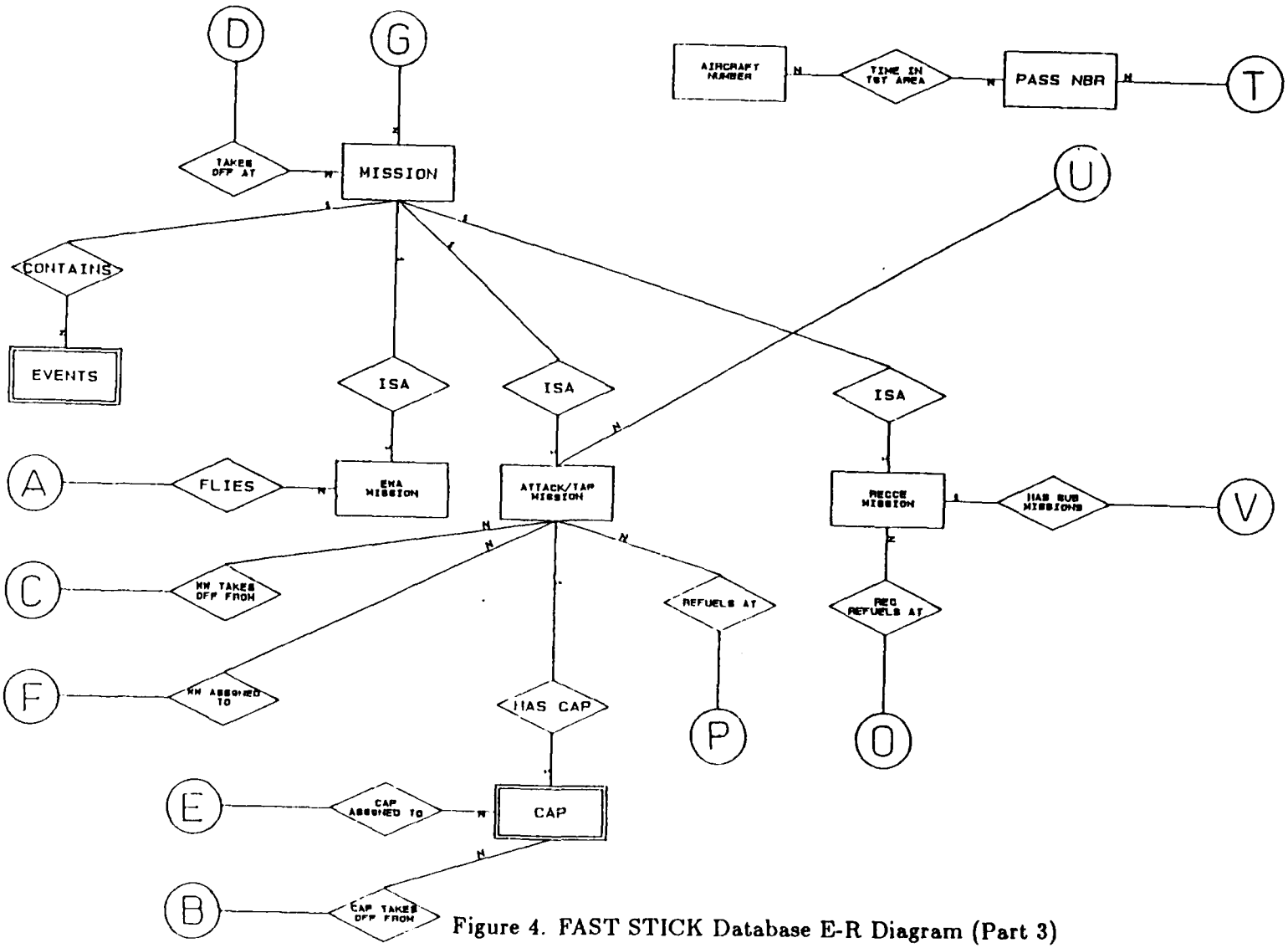

Figure 4. FAST STICK Database E-R Diagram (Part 3)

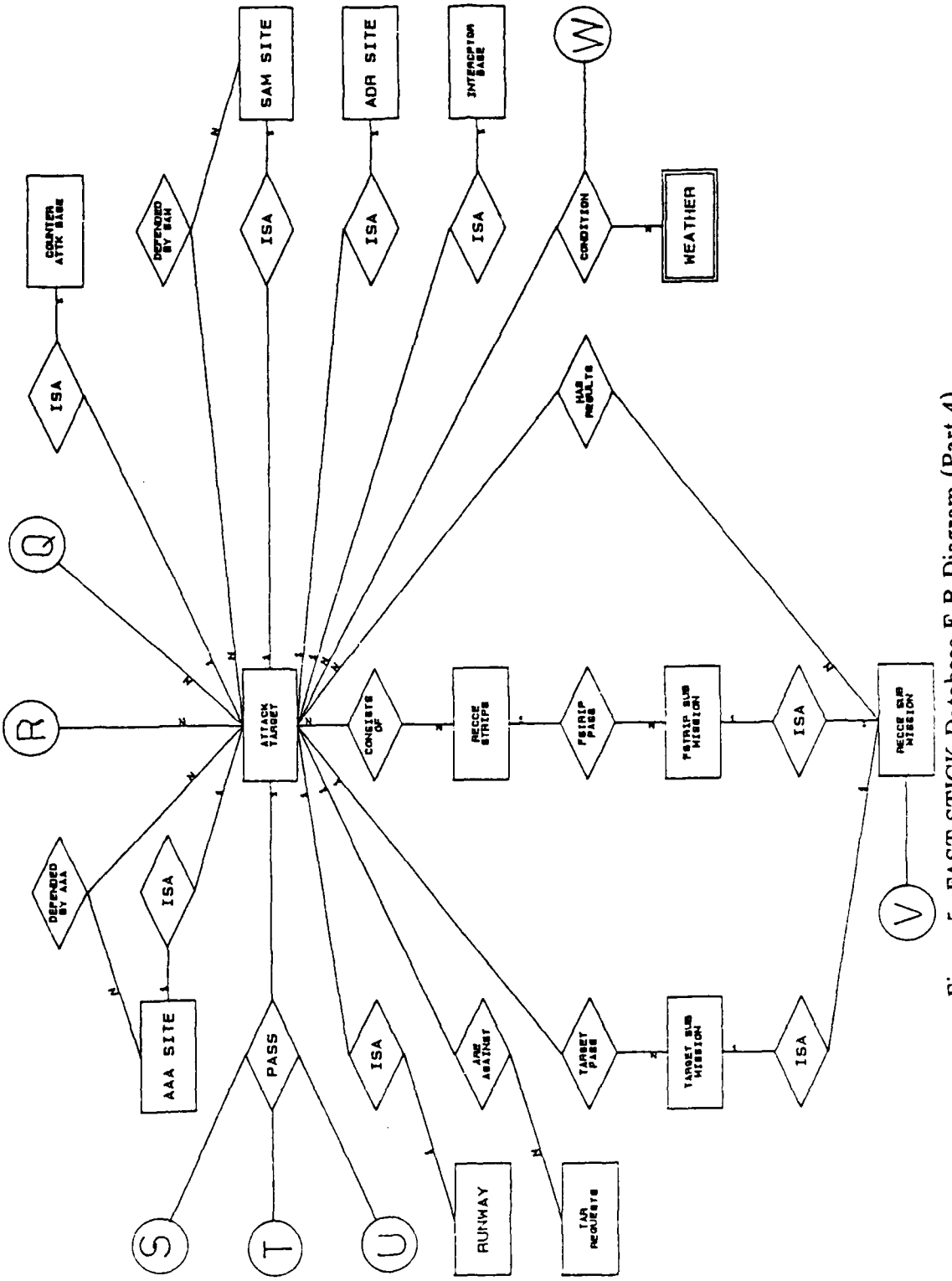

Figure 5. FAST STICK Database E-R Diagram (Part 4)

III. FAST STICK Database Implementation

This chapter describes the steps taken to implement the FAST STICK database, implement the automated data interface between JPLAN and FAST STICK, and replace the application flat file references in the simulation code with INGRES embedded database calls. The first section discusses how the FAST STICK E-R diagram was converted to candidate tables, followed by an examination of design issues to consider regarding implementation of a relational database. Next, the actual database implementation using the E-R Designer tool and the INGRES database management system is presented, including modification of storage structures in INGRES to optimize query performance. The fourth section describes how a simple user interface was developed for each of the tables in the database that held parameter or probability data using a tool provided by the INGRES development environment. The next section discusses how the data interface was established between JPLAN and FAST STICK to permit transfer of information. Finally, the last section discusses incorporating the INGRES database system into the simulation program and the problems encountered.

3.1 E-R Diagram to Table Conversion

After completing the E-R diagram, the next step was to convert the entities and relationships into candidate tables. Entities were directly converted into tables by simply placing the attributes into the columns of the tables. For relationships, the primary keys of the two entities which the relationship belongs to are placed as columns in the table along with any attributes associated with that relationship. In the case of weak entities, the table is composed of attributes from two entities, the weak entity and the strong entity it is dependent upon. Only the primary key attributes from the strong entity are used to construct the table. For transforming entities that represent specialization or generalization into tabular form, two different methods were considered. In the first method, a higher level entity is transformed into a table using the same technique described above for entities. The lower level entity is transformed into a table by using the descriptive attributes of the lower level entity along with the primary key of the higher level entity to form the

columns of the table. In the second method, the higher level entity is not transformed into tabular form. Instead, the attributes of the lower level entity and higher level entity are combined to create a table for each lower level entity. In this thesis effort, the first method was used. This method was selected because each of the higher level entities representing specialization in the FAST STICK E-R diagram were composed of many attributes. Had the second method been used, a large amount of data would have been dispersed in the tables representing the lower level entities of the specialization.

3.2 Relational Database Design Considerations

After the candidate tables have been constructed, a designer must consider design issues for implementation of a relational database. "The goal of a relational database design is to generate a set of relation schemas (tables) that allow the user to store information without unnecessary redundancy" [15:173]. This is normally accomplished by decomposing tables with a large number of attributes into several relations with fewer attributes. Constraints on the tables called data dependencies are used to determine how the tables will be decomposed. One particular kind of data dependency that is used is a functional dependency expressed as $X \rightarrow A$. The expression states that attribute Y is functionally dependent on X if each value of X determines a unique value of Y. The reason functional dependencies are used is to avoid certain undesirable properties that can occur in decomposition of tables [15:188]. In particular, the loss of information when a table is decomposed into two smaller tables and then reconstructed. Such a decomposition of a relation is referred to as a lossy join decomposition [15:178]. To ensure that relations do not have such anomalies, relations are normalized.

In normalization, a set of functional dependencies are used to decompose a table into a particular normal form. Normal forms represent the various degrees of redundancy that can be eliminated in a relation [15:192]. "A relation is said to be in a particular normal form if it satisfies a certain specified set of constraints" [9:362]. These normal forms are described in terms of stages referred to as first, second, third, Boyce Codd, fourth, and fifth normal forms (see Figure 6). At each successive stage, certain undesirable features are eliminated from the initial unnormalized table. "One of the more desirable normal

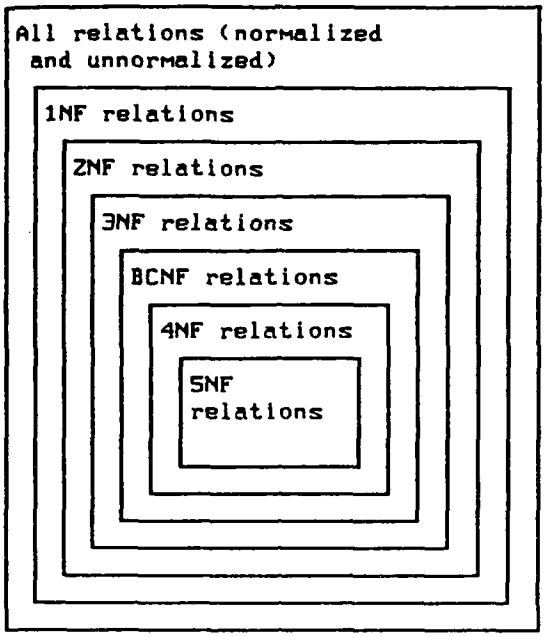

Figure 6. Different Stages of Normal Forms [9:363]

forms we can obtain is Boyce Codd normal form (BCNF)" [15:192]. A relation is in BCNF if for all functional dependencies that hold on the relation, at least one of the following constraints holds [15:192]:

1. $X \rightarrow A$ is a trivial functional dependency.
2. X is a superkey.

A trivial functional dependency is a functional dependency that is satisfied by all relations [15:183]. Formally, FD $X \rightarrow Y$ is trivial if Y is a subset of X [15:183]. Some examples of trivial FDs are: $A \rightarrow A$ or $AB \rightarrow A$ or $AB \rightarrow B$. A superkey is a set of one or more attributes (fields of a table) that ensures that each tuple (row) in a relation (table) is unique [15:49]. By ensuring that relations are in BCNF, data redundancy is minimized in a database while data integrity is maintained.

In general, "normalization ensures that relations are broken into simpler relations in which related data items are grouped, and duplication is minimized" [1:2-3].

3.3 FAST STICK Database Implementation

As stated in chapter 2, the E-R Designer tool stores all the information necessary for converting the diagram into a specific DBMS format in an internal dictionary. The E-R Designer package provides another tool called SchemaGen to convert these E-R Designer files containing the E-R diagram into database schemas specific to several commercial relational database systems. The INGRES data definition language format for generating database schemas is one of those systems supported by the tool.

The E-R Designer files of the FAST STICK diagram were entered as input to the SchemaGen tool. However, the tool was unable to convert the diagram into the INGRES format. It had problems with the entities involved in "ISA" specialization, and the one recursive relationship in the diagram. Whenever the tool came across either one of these cases, it would stop processing on the diagram and report an error. After consultation with the product's designers, the following information was obtained: all "ISA" relationships had to placed on the same line of the grid display. However, no explanation was given for the problem encountered with recursive relationships. The product's designer stated that they had fixed these problems in a new version of the tool that was currently in beta testing. The company sent this beta version to us. The new version was tried against the FAST STICK E-R diagram and the results were different. The error occurring with the "ISA" specialization had disappeared but the problem with the recursive relationship still persisted. In an attempt to get the tool working on the FAST STICK E-R diagram, the recursive relationship was removed from the diagram. The diagram was once again run through the tool. The result was another error caused by an "ISA" specialization associated with the *Recce Sub Missions* weak entity in the diagram. After several more days of attempting to resolve this problem with no success, it was decided that rather than attempting to debug this software it would be much easier to simply convert the FAST STICK diagram to table definitions manually and enter them using the implementation tools provided in the INGRES DBMS package.

Before the table definitions were entered, each candidate table was normalized into Boyce Codd normal form. This was accomplished by identifying the functional dependen-

cies in each of the tables and checking to make sure they met the constraints required for Boyce Codd normal form. All of the tables were in Boyce Codd normal form.

The next step in the implementation process was rather simple and mechanical. It involved using INGRES system-level commands and the menu driven development environment to create the database and enter the table definitions of each of the candidate tables. The FAST STICK database was created by using the system level command CREATEDB. This command creates the directories and system files needed to store and manage data in the database. Next, the INGRES menu driven development environment was executed. In the menu driven environment, the option TABLE was selected in order to display the Table Utility Screen. From this screen, the option CREATE was selected. The Table Create Screen then appeared prompting the user to enter the name of the table to be created, the name of the fields (columns) that comprise the table, and the data type for each field. As the information for each table was entered, the information was stored in the FAST STICK database. After entering all the table definitions, the FAST STICK database consisted of 96 different tables.

With all the tables defined for the database, the next consideration was the type of storage structure each table would use. INGRES provides different types of storage structures in order to allow a user to optimize the time required to access and update data in the tables [6:3–8]. On the PC version used for this thesis, the following structures were available [6:B–2]:

1. Compressed heap (Cheap) - a random, unordered storage structure in which duplicate rows are not removed and new rows are added at the end of a file with trailing blanks removed.
2. Compressed btree (Cbtree) - a compressed binary tree indexed on specific attributes with trailing blanks removed. All duplicate rows in the table are removed.

When tables are initially created the default storage structure is a compressed heap [6:B–2]. The compressed heap structure is better for small tables, and requires less overhead when addition or deletions are made because it is a simpler structure than Cbtree

with key based access requirements [6:3–8]. However, Cbtree is better for large tables [6:3–8]. In the FAST STICK database, all tables that contained probabilities and constants data, and had a large number of records were modified to Cbtree storage structures. The reason for selecting the Cbtree structure for these tables was because records within a Cheap structure are unordered, which means INGRES must search the entire table for each database query. This type of structure is not as efficient as Cbtree structure when retrieving data from a table using an indexed field. Since the tables in FAST STICK containing probabilities and constants data will be primarily retrieve only on certain key indexed fields, their structures were modified to Cbtree using the INGRES system level command MODIFY.

3.4 User Interface to FAST STICK Parameter and Probability Data

As stated in Chapter 1, the FAST STICK program used flat files to store the multitude of probabilities and parameters needed to simulate the various events that occur in the exercise. Using this type of storage method made it difficult for game controllers to change exercise data quickly and easily, especially in cases where they wanted to calibrate the exercise for different scenarios or learning objectives. This shortcoming was solved by using one of the integrated tools for end user decision support provided within the INGRES DBMS package. Using the integrated tool called Query-By-Forms (QBF), an end user interface was developed that allowed controllers to easily view and/or update the exercise data in the FAST STICK database. "Query-By-Forms is an interactive, visually oriented, forms based module for adding, deleting, changing, and viewing data in an INGRES database" [6:3–8]. With this tool, users could use forms to access and update data within a database. A form in INGRES is simply a large sheet of paper displayed in a electronic format on a computer screen. The form consists of various components that either display information or reserve space to enter information. Forms can be customized to display exactly the data the end user wants or needs to see from the database. By using a screen oriented editor called Visual Forms Editor (VIFRED), a developer can create a specific form on a screen and associate a particular table in a database with that screen. Using this capability, a customized form was created for each of the 28 tables that held

constants and probabilities data within the FAST STICK database. These forms permitted game controllers to go into the database and change exercise data without a great deal of effort.

3.5 Automated Data Interface

As mentioned in Chapter I, FAST STICK is the final step in a sequence of Air Command and Staff College joint planning exercises to deploy and employ air forces against the forces and targets of an imaginary enemy in a theater setting. FAST STICK simulates the employment half of the exercise while deployment planning of forces is simulated by another program called JPLAN. JPLAN is used in the exercise to plan the composition of combat forces, and build force lists. The results of the JPLAN program are used as input into the FAST STICK program. Currently, all data generated by JPLAN has to be manually transferred between the two systems by game controllers.

In a past thesis effort, the JPLAN exercise was redesigned to run on a microcomputer with the INGRES database management system and tool set [14:4]. Since JPLAN used the INGRES database system with its integrated tools, the necessary connections were there to allow JPLAN to interface with other exercises using INGRES. With this capability, an automated data interface was built between the two systems using the system level facilities and Structured Query Language (SQL) available in INGRES. The system level facilities used were utility programs provided by INGRES for creating, manipulating, and managing databases at the operating system level [6:5–1].

The automated data interface was developed by creating an MS DOS batch file containing DOS commands and INGRES system level commands that execute SQL command files. Figure 7 contains the commands found in the batch file.

The commands in the batch file execute the following series of operations:

1. Check if a table called *base_ac* exists in the JPLAN database. If it does exist then delete it from the JPLAN database.
2. Create the temporary table *base_ac* in the JPLAN database and then load it with data from the tables *air_tpfdd* and *tac_ac_utc*. The table *base_ac* consists of three

```
if exist c:\ingres\data\jplan\base_ac.Opc
    ingbatch -sql jplan <drop.bac
ingbatch -sql jplan <create.bac
ingbatch -sql jplan <update.bac
copydb jplan base_ac
quel jplan <copy.out
if exist c:\ingres\data\fastick\base_ac.Opc
    ingbatch -sql fastick <drop.bac
quel fastick <copy.in
ingbatch -sql jplan <drop.bac
ingbatch -sql fastick <delete.dpl
ingbatch -sql fastick <insert.bac
ingbatch -sql fastick <drop.bac
erase copy.out
erase copy.in
```

Figure 7. TRANSFER.BAT Batch File

fields: airbase, aircraft type, and number of aircraft of a particular type assigned to an airbase.

3. Update all instances of "PRM" in the airbase field to "PRI" as well as all instances of "EF111A" in the aircraft type field to "111A". The reason for this update is because the two exercises use different abbreviations for the airbase "Prima" and the aircraft type "EF 111A" in their databases.
4. Unload the *base_ac* table in the JPLAN database to the file *base_ac.Opc*.
5. Check if the temporary table *base_ac* exists in the FAST STICK database. If it does exist then delete it from the FAST STICK database.
6. Create the temporary table *base_ac* in the FAST STICK database and then load it with data from the file *base_ac.Opc*.
7. Delete the table *base_ac* from the JPLAN database.
8. Delete all data from the table *deployed* in the FAST STICK database.
9. Copy data from the *base_ac* table in the FAST STICK database to the *deployed* table.
10. Delete the table *base_ac* from the FAST STICK database.

11. Erase the INGRES command files, *copy.out* and *copy.in*.

By simply executing this one DOS batch file, data that was needed by the FAST STICK exercise from the JPLAN exercise was automatically transferred from the JPLAN database to the FAST STICK database. NOTE: The contents of each of the SQL command files displayed above can be found in Appendix B.

3.6 Simulation Conversion to Database System

Originally in this thesis effort, the INGRES database system was to be incorporated into the FAST STICK simulation program by having every application flat file reference replaced with an INGRES embedded database call, but because of two constraints in the current operating environment this was not feasible. Embedded database calls are SQL (Structured Query Language) commands embedded within programs written in another programming language [5:1–3]. They permit the database to be accessed by an executing program.

The first constraint encountered dealt with main memory and the MS DOS operating system environment. The Zenith 158 microcomputer running MS DOS has 640 kilobytes of internal memory available. Approximately 62 kilobytes of this memory is used by the MS DOS operating system. The INGRES database management system requires approximately 220 kilobytes of code to remain resident in main memory on a IBM compatible microcomputer. The FAST STICK simulation program with embedded database calls requires approximately 320 kilobytes of memory at the beginning of execution. However, the program may require additional memory as it is executing. Exactly how much is dependent on the number of missions being simulated. With this large amount of memory being utilized, it is very possible for the simulation program to run out of memory while simulating a large number of missions.

The second constraint was the slow access time of the Bernouli disk on the Zenith 158 microcomputer. The simulation program read and wrote large quantities of data to and from secondary storage (disk). Using the random access methods of the original application flat file system, the slow performance of the disk was not extremely noticeable. However,

when data was accessed through the Bernouli disk using the INGRES embedded database calls, the slower performance was very noticeable. Retrieval and storage of exercise data using the INGRES database management system coupled with the Bernouli disk would have resulted in poor run time performance of the simulation program [7]. Additionally, the INGRES database management system for IBM compatible microcomputers only permitted database calls to be embedded within C programs. The FAST STICK simulation program was written in Pascal. In this situation, the Pascal simulation program would have had to call a C subroutine to access the database. This would have required additional run time for conversion of C data variables and structures to Pascal data variables and structures.

These constraints are not a result of the INGRES DBMS but rather a result of the system environment in which it was required to operate. To circumvent these constraints and still use the INGRES DBMS, it was decided to continue to run the FAST STICK program with flat files, and also have a copy of the exercise data in an INGRES database. Game controllers would access, manipulate, and add data through the INGRES Query-By-Forms tool. The data in the database would be down-loaded to the flat files to run the simulation.

In order to down-load the data, a series of conversion routines were written that accessed the database through embedded calls and converted the data into the application flat file record format. These conversion routines were linked together by a driver routine which allowed the game controller to select which flat files he wanted updated from the database. The driver routine provided a menu driven screen to make selections. A copy of this screen display can be found in Appendix C.

IV. Land Battle Simulation Addition

This chapter addresses the issues concerned with the addition of land play to the exercise. The first section of this chapter is an introduction to combat modeling concepts. It is not intended to be a comprehensive review of combat modeling, rather it attempts to briefly examine several high level aspects of combat modeling the author considered in developing the land simulation play. The next two sections provide a brief description of how the FAST STICK exercise is played and the FAST STICK computer model. The subsequent sections address the land simulation extension to the FAST STICK computer model and its' implementation. The addition of land play had to be accomplished with minimal impact on the rest of the simulation program. This was requested by the sponsors of this thesis. Using this strategy to add land play greatly assisted in the completion of this part of the effort because of the size and complexity of the simulation program. As it turned out, adding the land simulation to the exercise was the most difficult part of this thesis project. The design of the extensions to the computer model and implementation required over seven weeks of effort to complete. The final section of this chapter discusses the development of a scenario generation input application for entering land battle scenario data into the FAST STICK database.

4.1 Introduction to Combat Modeling

The military has a unique problem in that it deals in a profession that does not always have the luxury of training or educating its members in a realistic setting or environment. This is especially true in instances where officers are being educated in new doctrine and concepts of warfare. The cost and possible threat to life associated with live military exercises prohibits their use as a general educational tool. Instead, models are built to simulate combat environments. These models combined with digital computers can be used to examine the complex issues and concepts found in combat decision making environments without the drawbacks of live exercises.

In the broadest sense, modeling is the process of designing a mathematical-logical model of a real world system and experimenting with this model on a computer in order

to study the output of such real world systems under various conditions [16:6] or [12:I-1]. Models normally are built to represent an abstraction of some small portion of reality [8:7]. In modeling combat, the basic combat processes of maneuver, target acquisition, target engagement, command and control, communications, and resupply are being simulated.

Combat models are used in training and education "to reinforce desired lessons" [10:7]. "They create a simulated combat environment in which specific military decision tasks can be practiced and evaluated" [12:I-12]. "Models which allow human interaction are particularly useful for training purposes" [12:I-12].

There are basically two types of combat models, high resolution and low resolution (aggregated) [11:I-2]. The classifications of the combat models tend to be deceiving and imply the opposite of what they actually stand for. In a high resolution combat model, a detailed view of warfare is provided whereas in low resolution details about individuals and individual engagements are left out [12:I-7]. A high resolution combat model is one in which the basic model entities are individual combatants or weapon systems [12:I-6]. "Each such entity has numerous attributes which define its unique position in the force, its unique perception of the battlefield and the enemy force, its capabilities, and its activity at each moment of simulated battle time" [11:I-2]. A low resolution combat model is one in which the basic model entities are groups rather than individual combatants [11:I-6]. Simulation entities in the low resolution models are combat and support units where as in the high resolution models the entities are individual combatants [11:I-7]. Overall, the difference between the two types of models is that low resolution models simulate combat using "average behavior" while high resolution models simulate combat using "individual averages" [11:I-8].

There are basically six techniques used to model general purpose forces such as those found in the FAST STICK exercise [8:75]. These techniques can be used with either low or high resolution models. One such modeling technique is the construction of simulation models. Simulation models are detailed mathematical representations of real world situations used where physical events are well understood [8:76]. In these types of models, probability distributions are associated with each of the physical events represented in the model. These values are representative of actual combat parameters [8:76]. A Monte

Carlo method is used to generate random numbers that are compared against the event probability distribution to determine the outcome of an event.

The next type of modeling technique uses differential equations to model the concentration of firepower in combat. These types of models are referred to as Lanchester models of combat. In these models, differential equations are used to represent the strengths of the two forces in conflict. The losses each side sustains per unit of time during combat is directly proportional to the numerical strength of the opposing force [8:89].

A third type of modeling technique, Firepower Score models, is concerned with the interaction of firepower between opposing forces. Each force being modeled in an engagement has a firepower score or potential. "The firepower score of opposing units is determined as the sum of a linear combination of individual weapon firepower scores times the number of weapons of each type in the unit" [8:104]. "Firepower scores have been derived based on the estimated casualty producing potential of each weapon type" [8:104]. The outcome of engagements is determined by comparing the force ratio of the two forces with predetermined tabular values [8:104].

In a slight derivation from the Lanchester model described above, Heuristic models also use differential equations to describe attrition and forward edge of the battle (FEBA) movements. However, this type of model uses equations that are heuristic in nature. "The equations are asserted by experienced analysts as being the reasonable representations of the results of large scale interactions but with no direct historical or analytic derivation" [8:114].

The final two techniques, Allocation models and Force Structure Phasing models examine specific areas within general purpose force modeling. They are concerned with optimum allocation strategies and long term planning of force structures. Each of the techniques can be considered a hybrid of any of the above models.

After review of the above modeling techniques, it was decided that none of the them were completely appropriate for adding land play to the FAST STICK exercise. Rather, another technique that was somewhat similar to simulation modeling was developed that

modeled land battle through the use of scripted battle scenarios. A scenario consisted of a sequence of events that described battle actions through text.

From the broad usage of models in the military, it seems that modeling has achieved a level of respectability in terms of accurately representing combat in the real world. However, combat models do have limitations and are often misapplied as educational tools. One of the dangers of using models in an educational environment is that students may carry away the wrong lesson [10:50]. Simulation models typically reward adherence to current employment doctrine therefore discouraging students from trying new strategies or tactics [10:50]. Additionally, models may be used in a different context than for which they were originally designed. "Models in general have been tailored with specific situations in mind, and hence any attempt to use the model in a different context will require great scrutiny of basic modeling assumptions, model logic, and the nature of inputs" [8:7]. With these limitations in mind, the game controllers of this exercise must realize that enhancements to the FAST STICK game in terms of land play must be taken in the context for which they were developed. The fact that the exercise can generate different scenarios to teach different learning objectives and concepts must be closely scrutinized to insure that students are learning the correct lessons and concepts. Further discussion of the flexibility that the design offers in terms of generating different scenarios will be addressed later in this chapter.

4.2 FAST STICK Exercise Description

The FAST STICK exercise attempts to simulate the environment to which an individual would be exposed to in the plans and operations branches of a Tactical Air Control Center (TACC) during a tactical war. In the exercise, students are members of a team that make up the TACC branches. Each team member performs a staff function of one of these branches. During the exercise, team members determine the priority of targets to be destroyed, assign a desired damage expectancy for each target, plan reconnaissance missions to obtain more information, and then decide on which targets to attack.

Before the game is started, the students are briefed on the general scenario of the game. The scenario basically depicts two fictitious countries with equal offensive and

defense capabilities with open hostilities. The computer game is programmed with these capabilities and represents one of these countries, while a team of students represents the other. A team starts with a limited number of aircraft resources and is expected to meet stated objectives from higher command directives.

The FAST STICK simulation is played over four calendar days, while the game itself simulates only three days of actual events. On the first calendar day, a team will plan and conduct reconnaissance missions to obtain more information on enemy targets. On the second calendar day, a team reviews the reconnaissance data from game day one to plan attack and reconnaissance missions for game days two and three. Game days two and three are played on the third and fourth calendar days. On these days, a team plans and conducts attack and reconnaissance missions. At the end of the fourth day, the FAST STICK program compiles a score for the team based upon the number of targets destroyed, and the remaining aircraft resources.

All planning and flying is based on two cycles: morning and afternoon. A typical game day computer sequence of events would occur as follows [2:3-3]:

Morning Cycle

1. Logon to the game.
2. Reserve air defense aircraft.
3. Reserve spare aircraft.
4. Reserve close air support aircraft.
5. Input flight plans with takeoff times from 0001 to 1159.
6. Engage simulation.
7. Receive results from flights that recover prior to 1200.

Noon Take 30 to 40 minutes to assess the morning results and make any changes or additions to afternoon flight plans.

Afternoon Cycle

1. Logon to the game.

2. Reserve air defense aircraft.

3. Reserve spare aircraft.

4. Reserve close air support aircraft.

5. Input flight plans with takeoff times from 1201 to 2359.

6. Engage simulation.

7. Receive results from flights that recover prior to 2400.

At the end of the day, the program provides a summary of the day's activities.

4.3 FAST STICK Computer Simulation Model

The FAST STICK simulation is event driven. An example of an event would be an aircraft taking off from an airbase. The computer model for the game is based on a set sequence of events occurring during a mission. For each type of mission found in the simulation program, a predetermined sequence of events exists. Each of these events has either a fixed or variable probability of success. In the model, the probability of success of an event is compared against a random number generated between zero and one by a Monte Carlo random number generation process. If the number generated is less than or equal to the probability of the event, the event is a success, and the program moves to the next event in the mission sequence. In order for a mission to be successful each event in the mission sequence must be successfully completed.

There are basically two types of missions found in the model, reconnaissance and attack. However, deviations from planned events within these two types of missions are possible. The difference between the two lies primarily in the number of events and the variable probabilities of success. Attack missions for the most part have more events in their mission sequence with a considerable number of these events having variable probabilities of success. This is a result of the fact that attack missions may use additional support forces.

An example reconnaissance mission sequence is shown in Figure 8. In the diagram, the first event that occurs on a reconnaissance mission is takeoff. The .95 in the box

represents the probability of a successful takeoff. Should the takeoff be unsuccessful the aircraft is not damaged but will be unavailable for eight hours due to aircraft maintenance. The next event that may occur in the mission is air refueling. This event is dependent upon whether a team has scheduled refueling in their flight plan. If air refueling was scheduled then the probability of successfully refueling is .98. If air refueling was missed then the aircraft attempts to return to base, otherwise the aircraft proceeds on with its mission. The next event to occur in the mission sequence is the aircraft flying over the target while taking photographs. The probability of the reconnaissance flight surviving mission sequence is the aircraft flying over the target while taking photographs. The probability of the reconnaissance flight surviving while over the target is .98. Should the aircraft be hit by enemy defenses while over the target, it will attempt to return to base and land. The probability of a successful landing in such a case is .60.

If the aircraft lands, it will undergo maintenance. In the maintenance event, the aircraft will have a .10 chance of being in maintenance for 8 hours, a .40 chance of being in maintenance for 24 hours, and a .50 chance of being permanently damaged. Should the aircraft fail to land it is considered destroyed. If the aircraft was not hit while photographing the target, then the next event determines whether useful intelligence information was photographed. This event has a variable probability based on the weather conditions over the target. As the aircraft begins its return trek, it may be scheduled for air refueling. This is dependent on a team's flight plan. The air refueling event has the same probability of success and follows the same sequence as mentioned above. However, should the aircraft miss refueling with a tanker, the aircraft may not have enough fuel to return to base. If the aircraft cannot reach its' home base or a friendly base with its' remaining fuel, it crashes. Following refueling, the activity of returning to base and successfully landing is the next event. If this event is unsuccessful the aircraft is considered destroyed. The final event in the reconnaissance event sequence is a maintenance check to determine whether or not the aircraft will remain in commission and be available for additional missions. The probability of success is .80. Should the aircraft fail the maintenance check, it has a .40 chance of being in maintenance for 8 hours or a .60 chance of being in maintenance for 24 hours.

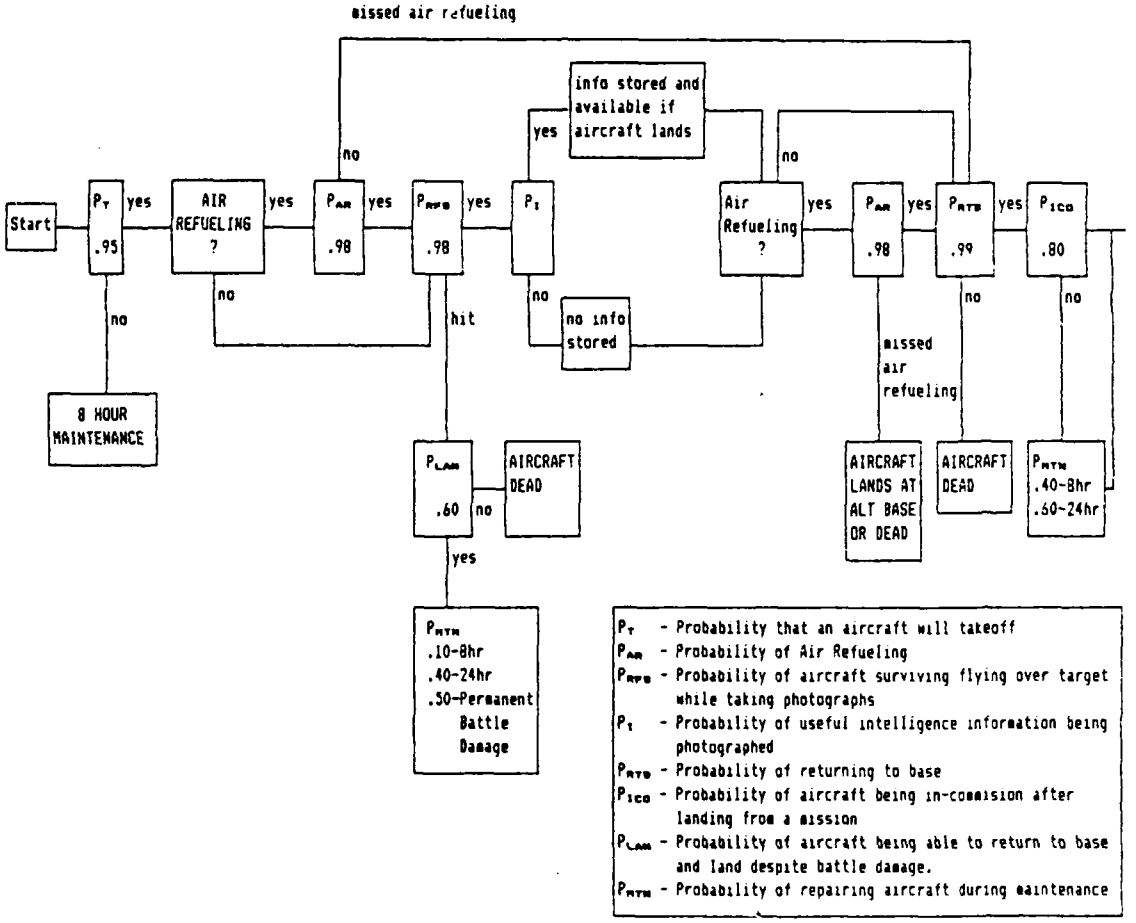

Figure 8. Reconnaissance Mission Simulation Sequence

4.4 Land Battle Model

As discussed in chapter one, FAST STICK has no land or battle simulation. However, one of the objectives of this exercise is to teach Air Force officers how to apply close air support in a joint combat environment. It states in the FAST STICK users manual that ground actions are taking place at the same time that the air war is going on. However, the current implementation of the exercise does not display or include any type of ground action events. The only type of action involving close air support is random generation of requests for close air support (CAS) by the simulation. The only results players see is whether or not they have destroyed the CAS target. They are unable to observe the impact that their CAS allocation decision has on the outcome of a battle in a particular scenario or setting.

In the current exercise, the CAS requests generated by the simulation have no relevance to any type of concerted effort on the part of either side in the conflict to achieve some tactical land or battle objective. The importance of supporting requests for close air support is emphasized in the exercise by penalizing a team 500 points for not responding. Although this is adequate in terms of ensuring that a team responds to such requests, it does not truly emphasize to players the important role that close air support can play in a joint operation. If players were in a real world situation such as portrayed in FAST STICK, they would most likely be informed or at least aware of the ground actions occurring in the conflict. Therefore, this area of the exercise is somewhat lacking in providing a realistic setting for a Tactical Air Control Center. The need exists in this exercise for some type of mechanism to inform players of current ground actions occurring in the theater and the effect that the close air support missions they allocate will have on the outcome of battles in the theater.

In this thesis, this need was addressed by adding scripted battle scenarios containing land battle events to the exercise. From analysis of this exercise, it was determined that the best means of adding land play was not to create a separate land battle simulation that ran by itself, but to provide game controllers with a tool that would allow them to generate different battle scenarios so that they could emphasize different aspects of close

air support in different settings. Such a tool would allow them to change scenarios as close air support concepts and doctrine were updated or modified.

In order to accomplish this, an extension was added to the current FAST STICK computer model so that land battle events could be included. The land battle scenario generation extension requires the game controller to design a particular series of ground actions that would occur in the exercise. The ground actions would consist of descriptions of battle events that could occur at a particular time in the exercise. A description of a battle event might state that a Brazonan unit was attacking an Iguanan unit with armor and heavy artillery. Follow on battle events would describe other actions occurring in the battle, the direction the battle was going, losses each side was taking, etc. This information would be displayed on the screen to players as the events occurred during the simulation run. At certain points during the battle, a request for a close air support would occur. Players would respond to requests by allocating a certain number and type of CAS aircraft to this request. The CAS mission would then be simulated. Depending upon the number of aircraft they assigned, the generation of a random probability, and whether the target was destroyed or not would determine the next sequence of battle events. The player would see the impact of his CAS allocation decision in the next series of battle events that occurred.

There were two reasons why it was decided to simulate ground actions in this particular manner. First, this method provided flexibility to the exercise in terms of allowing the game controllers to generate different settings or situations in which to teach different doctrine or aspects of close air support. Second, this method had very little impact on the rest of the simulation program. The Air Force Wargaming Center had requested that other parts of the simulation program not change or be affected in order to avoid the exercise being radically different from its current state. They were concerned that a radical change might result in the exercise no longer meeting the same educational objectives.

As stated above, the extension to the FAST STICK model basically consists of the addition of a sequences of battle events occurring at specified times. Whenever a CAS event occurs, the result of the CAS mission determines the next sequence of battle events. The following factors influence the result:

1. Number of CAS aircraft assigned to the close air support request.
2. Whether the CAS target was destroyed or not.
3. The result of a random number function generating a number between 0 and 1.

Figure 9 on the next page is a graphic representation of events in a scenario. The squares represent land battle events that will occur on a particular day of the exercise. The circles represent requests for close air support that will become CAS mission events, while the diamonds represent the possible paths the scenario can take after the CAS mission is completed. The numbers in each of the geometric shapes specifies an event number. These numbers are used to uniquely identify each event in a scenario. Associated with each event type in the scenario diagram is a form where specific event data is recorded (See Figures 10, 11, 12). The game controller uses the combination of the scenario diagram and event data forms to design battle scenarios to be entered into the simulation program.

4.5 Land Battle Simulation Implementation

The addition of land simulation was accomplished by adding two new event types to the simulation program. These two event types represent battle events and decision events. Battle events in the simulation are text descriptions of events occurring in the ground battle scenario designed by the game controller. Decision events follow the completion of close air support missions. They are used to check the results of such missions to determine the next sequence of scenario events to be simulated. For each of these event types, Pascal routines were developed that simulate the events. The battle event Pascal routines stop execution of the simulation program, retrieve the text description of a battle event from a data file, and then display the text in a window on the screen of the Zenith 158. The student must then press the space bar for the simulation program to continue execution. The decision event routines added to the exercise examine data associated with the CAS mission previously simulated to determine the next sequence of scenario events to be simulated. In particular, the routines check the following information associated with a CAS mission:

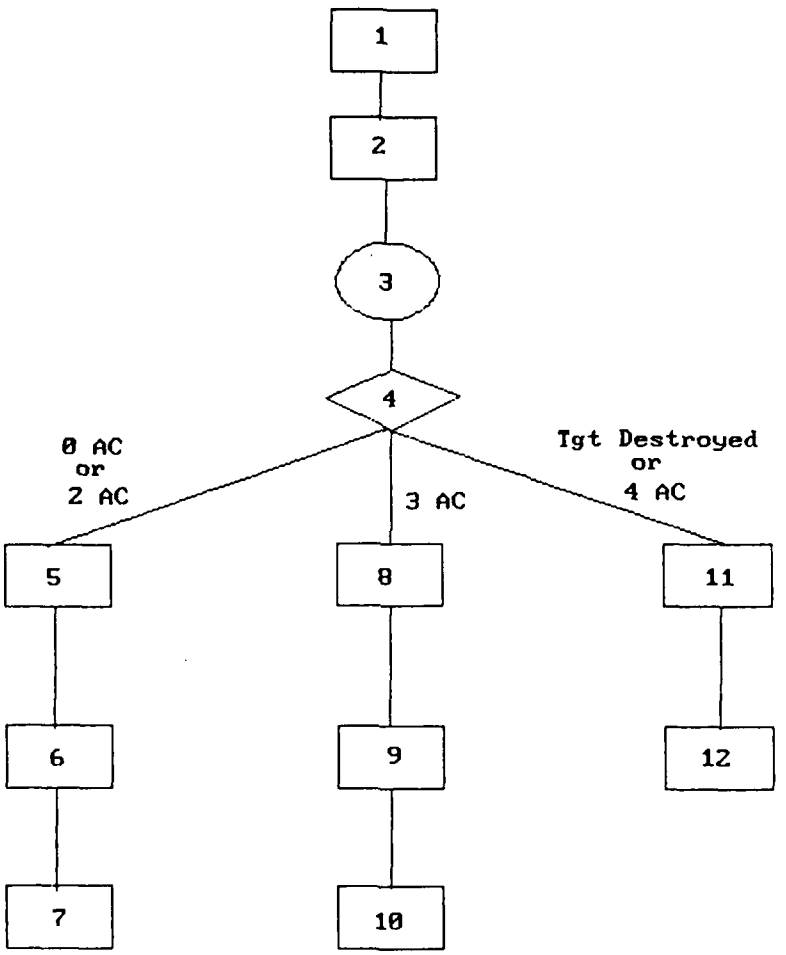

Figure 9. Graphic Representation of Scenario

Event Nbr	Game Day of Event	Game Time of Event	Description	Next Event

Figure 10. Sample Battle Event Data Form

Event Nbr	Game Day of TAR	Game Time of TAR	Max Takeoff Time for TAR Flight	TAR Target	Next Event

Figure 11. Sample TAR (CAS) Event Data Form

Event Nbr	Game Day of Event	Game Time of Event	TAR Event Nbr	Tgt Destroyed	AC Input	Probability	Next Event
				NO	0	1.0	
					2		
					3		
					4		
				YES	N/A	1.0	

Figure 12. Sample TAR Decision Event Data Form

1. The number of aircraft the student has allocated for the CAS.
2. The current status of the CAS target, destroyed or not.

The decision routines use this information along with the generation of a random number to determine the next sequence of scenario events that will be simulated in the exercise. The sequence of events that follow a decision event can consist of any of the following combination of events:

1. All battle events.
2. Battle events followed by a CAS event, followed by a decision event.
3. A CAS event followed by a decision event.

4.6 Scenario Generation Input Application

After game controllers have designed their scenario on paper, the next step is to enter the scenario data into the data files of the FAST STICK exercise. This was accomplished by developing an application to input such data using the Application-By-Forms feature of

the INGRES package. This feature is an application development tool which integrates a forms editor, fourth generation language, and source code manager into a total application development environment [4:1-3]. The data entered through the application was stored in tables of the FAST STICK database. This data was then downloaded to flat files using the conversion programs mentioned in chapter three.

The scenario generation input application is composed of a main menu and 3 data input subscreens. From the main menu, the game controller can select a subscreen for any of three types of event data he wishes to enter into the FAST STICK database. The three subscreens permit input of scenario data for battle events, close air support requests, and decision events after a close air support mission has been simulated for a close air support request. Within each of the subscreens, various checking is done to ensure that the scenario data entered for the different types of events is valid. The main menu and three subscreens can be found in Appendix D.

V. Conclusion and Recommendations

5.1 Conclusions

This thesis has described the FAST STICK exercise and how it is used as a teaching vehicle at the Air Command and Staff College to allow students to learn and apply the basic tactical employment concepts of air superiority, interdiction, close air support, and reconnaissance. The FAST STICK exercise was originally a manual war game. During the mid 1970s, a computer version of the game was developed. This early computer simulation program continued to be used in the theater warfare phase of the Air Command and Staff College curriculum until 1987. At that time, the exercise was redesigned and rewritten to incorporate new advances in computer software and hardware technology. The old version of the exercise was moved from a Honeywell mainframe to an IBM compatible microcomputer. The new simulation program was written in Pascal, and the user interface was replaced by a screen-oriented menu driven system. Although major improvements were made to the simulation program, the exercise was still constrained by the fact that it did not use the most modern software capabilities for data storage; rather, it continued to use the same storage method available at the time of its' original development. Additionally, the exercise was now being used to educate individuals in operating in a joint combat operations environment. The exercise had not originally been developed with this environment in mind; therefore, in the area of the application of air support to ground actions, the exercise was limited.

The goal of this thesis effort was to enhance the FAST STICK exercise by addressing these shortcomings. The shortcomings were overcome by accomplishing the following objectives:

1. Designing and implementing a relational database for the exercise data.
2. Developing database to file conversion programs.
3. Automating the data interface between JPLAN and FAST STICK.
4. Designing and implementing an event driven land battle scenario generation model.

5. Documenting system design and installation.

In order to meet the first objective, a preliminary analysis was accomplished on the current FAST STICK exercise to determine what the data in the files represented and was used for. During the analysis, current literature on database design methodologies was reviewed to determine a design approach for implementation. The top down design approach was selected. This approach develops an overall logical structure for a database by modeling the data and the relationships between data in a diagram. An automated structured diagrammatic tool called ER-Designer was used to generate the diagram. The diagram was then implemented using the INGRES relational database management system (DBMS).

Once the FAST STICK database was designed and implemented, an automated data interface had to be written to transfer combat forces data between JPLAN and FAST STICK. JPLAN is a computer simulation program used to do the deployment planning for FAST STICK. Using the system level facilities and structured query language (SQL) available in INGRES, a MS DOS batch file containing DOS commands and INGRES system level commands was created to accomplish the transfer. The automated interface saved game controllers time they normally had to spend manually transferring information between the two exercises.

The next objective was to replace the application dependent flat file code in the simulation program with INGRES embedded database calls. At this juncture of the thesis effort, two problems were encountered. The first problem was with memory on the IBM compatible microcomputer. The combination of the size of the FAST STICK simulation program and the amount of memory resident code involved in using the INGRES DBMS resulted in the possibility of the microcomputers memory boundary being exceeded. The second problem was with the slow access time of the Bernouli disk drive on the Zenith microcomputer. Although exceptable when using flat files, it becomes exceedingly slow when coupled with the INGRES database management system. Even if the memory problem had been resolved, the slow access time of the Bernouli disk coupled with the INGRES DBMS would have severely degraded the run time performance of the simulation program.

To solve these two problems, it was decided that the FAST STICK simulation program would continue to use flat files. Game controllers would view, manipulate, and add exercise data to a database through the INGRES database management system. The data in the database would then be down-loaded to the flat files through conversion programs. Conversion programs were written for each of the constant and probability files.

The last objective of this thesis was to add some type of land play to the FAST STICK exercise. Several different combat modeling techniques were reviewed to determine which technique was appropriate for the exercise. A simulation modeling technique using a Monte Carlo random number generation process was selected. This technique involves providing a detailed representation of the actual sequence of physical events that occur during combat. It was selected because of its flexibility to change and compatibility with the current exercise. Routines were added to the FAST STICK program such that the battle events of a pre-designed scenario were simulated during the exercise.

Finally, a FAST STICK scenario generator input application was developed using the INGRES Application-By-Forms application development tool. This application provided a user friendly interface from which game controllers could input and maintain land battle scenarios to be simulated in the exercise.

Overall, the environment the game controllers have to operate in to run the FAST STICK simulation program has improved. By enhancing the controllers access to the exercise data through the use of a database management system, automating the transfer of data between JPLAN and FAST STICK, and adding land battle events to the simulation, the exercise can now be easily modified or tuned to meet changing learning objectives and doctrine in the area of air combat operations in a joint military operational environment.

5.2 Recommendations

As mentioned in Chapter IV, several problems were encountered during the flat file to database conversion process and while adding land play to the exercise. These problems were not a result of using the INGRES database management system or the addition of land simulation. They were a result of the operating system environment in which the exercise

is currently operating. One of the following options should be considered to alleviate the system environment constraints:

1. The FAST STICK exercise should be ported to an IBM compatible microcomputer utilizing the Intel 80286 or 80386 microprocessor running the OS/2 operating system. The Bernouli disk drive should be replaced with a 20M hard drive to speed up disk access time.
2. The exercise should be ported to a workstation (e.g. DEC VAX Workstation or SUN 386i Workstation) where memory constraints are not a problem, and the INGRES database management system is supported.
3. The simulation part of the exercise should be ported to a mainframe computer that supports the INGRES database management system while the user interface should continue to reside on an IBM compatible PC/XT. The mainframe computer would act as a database server and simulation host, and would communicate with the IBM compatible PC/XT through the INGRES network software.

Once the operating system environment constraints have been removed, the application dependent flat files should be replaced by INGRES embedded database calls to the FAST STICK database developed in this thesis effort. Additionally, the use of indexes and other optimization techniques used with database management systems should be examined to accelerate data retrieval.

After the transfer of the exercise to a different operating system environment, the INGRES report generation tool should be used to develop the daily summary reports and end of game reports currently generated in the FAST STICK exercise. The routines in the simulation program that currently serve this function should then be removed. Finally, after all the above extensions and enhancements have been made, a follow on thesis incorporating real-time graphical displays of the events occurring during the simulation should be considered. The addition of graphics would greatly enhance the teaching potential of this exercise.

Appendix A. *Tables Derived from E-R Diagram*

The tables in this appendix are derived from the FAST STICK E-R diagram in Chapter II. These tables were created in the FAST STICK database.

ATTRIBUTES	*FORMAT*
Flight Call Sign	C-4
Game Cycle	I-1
Nbr of Attack Aircraft	I-1
Type of A/C to be used	C-4
Takeoff Time	I-4
Next Event	I-2
Time out of fuel	I-4

Table 2. Mission Entity

ATTRIBUTES	*FORMAT*	
Base	C-8	*

Table 3. Airbase Entity

ATTRIBUTES	*FORMAT*	
Type of Aircraft	C-8	*
Near Flight Time Limit	I-3	
Far Flight Time Limit	I-3	

Table 4. Aircraft Type Entity

ATTRIBUTES	*FORMAT*	
Type of Weapon	C-3	*
Attack Profile	C-7	

Table 5. Weapon Types Entity

ATTRIBUTES	*FORMAT*	
Type of Target	C-1	*
Target Type Description	C-34	

Table 6. Target Type Entity

ATTRIBUTES	*FORMAT*	
Type of Weather	I-2	*

Table 7. Weather Types Entity

ATTRIBUTES	*FORMAT*	
Team Number	I-2	*
Password Day 1	C-4	
Password Day 2	C-4	
Password Day 3	C-4	
Password Day 4	C-4	
Game Cycle	I-1	
Game Day	I-1	
Checkpoint	I-1	
Random Number Seed	I-1	
Next Event	I-2	
Debug Mode	I-1	
Strike Flag	C-4	
Event Counter	I-3	

Table 8. Team Entity

ATTRIBUTES	*FORMAT*	
Pass Nbr	I-1	*

Table 9. Pass Number Entity

ATTRIBUTES	*FORMAT*	
Nbr of Bombs	C-1	*

Table 10. Bombs Entity

ATTRIBUTES	*FORMAT*	
Air Refueling Area	C-7	*
Flight Currently Refueling	C-4	
Flight Currently Waiting	C-4	

Table 11. Booms Entity

ATTRIBUTES	*FORMAT*	
Type of Recce Target	I-2	*

Table 12. Recce Target Type Entity

ATTRIBUTES	*FORMAT*	
F Strip Number	I-2	*
F Strip Busy	L	*
In Region	C-1	

Table 13. Recce Strips Entity

ATTRIBUTES	*FORMAT*	
Type of Recce Target	C-1	*

Table 14. EW Track Entity

ATTRIBUTES	*FORMAT*	
Target Region	C-1	*
EW Aircraft Coverage	I-1	

Table 15. Target Region Entity

ATTRIBUTES	*FORMAT*	
Weapon Load Number	I-2	*
EWPOD Used	C-3	

Table 16. Weapon Load Entity

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*

Table 17. EWA Mission Entity

ATTRIBUTES	*FORMAT*	
TAR Weapon Load Nbr	I-2	*
Game Cycle Entered	I-1	

Table 18. TAR Load Entity

ATTRIBUTES	*FORMAT*	
Name of Fixed Probability	C-30	*
Probability	F-3	

Table 19. Fixed Probability Entity

ATTRIBUTES	*FORMAT*	
Game Day	I-1	*
Number of CAP Intercepts	I-3	*
Launch Air Defense Aircrft	L	
Vertically Dispersed CAS AC	L	
Vertically Disperse IC AC	L	

Table 20. Flight Info Entity

ATTRIBUTES	*FORMAT*	
TAR Request Number	I-2	*
Notification Time	I-4	
Max Takeoff Time	I-4	
Flight Call Sign	C-4	
Nbr of Aircraft Allocated	I-1	
Target Selected	C-3	
Team Response to Request	C-5	
Penalty Points for Refusal	I-2	
TAR Request Used	C-5	

Table 21. TAR Requests Entity

ATTRIBUTES	*FORMAT*	
Game Day	I-1	*
Aircraft Killed by CAP	I-3	
TAR Penalty Points	I-4	
Target Hit Points	I-4	
Enemy Aircraft Killed Pts	I-4	
MisMgt of AC Penalty Pts	I-4	

Table 22. Totals Entity

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*

Table 23. Recce Mission Entity

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*
Nbr of WW Aircraft	I-1	
EW Pod	L	

Table 24. Attack/Tar Mission Entity

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*
Nbr of CAP Aircraft	I-1	
Type of A/C to be used	C-4	*

Table 25. CAP Weak Entity

ATTRIBUTES	*FORMAT*	
Target Number	I-3	*
Game Cycle	I-1	*
Hour within Game Cycle	I-2	*
Weather Condition	I-1	

Table 26. Weather Weak Entity

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*
Event Time	I-4	
Event	I-1	*

Table 27. Events Weak Entity

ATTRIBUTES	*FORMAT*	
Target Number of AAA	I-3	*

Table 28. AAA Site Entity

ATTRIBUTES	*FORMAT*	
Target Number of AAA	I-3	*
Target Number	I-3	*

Table 29. Defended By AAA Relationship

ATTRIBUTES	*FORMAT*	
Target Number of SAM	I-3	*

Table 30. SAM Site Entity

ATTRIBUTES	*FORMAT*	
Target Number of SAM	I-3	*
Target Number	I-1	*

Table 31. Defended By SAM Relationship

ATTRIBUTES	*FORMAT*	
Target Number	I-3	*
Air Defense Site Nbr	I-2	

Table 32. ADR Site Entity

ATTRIBUTES	*FORMAT*	
Number of AC Over Target	I-1	*

Table 33. Aircraft Number Entity

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*
Aircraft Tail Number	I-3	*

Table 34. Assigned To Relationship

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*
Refueling Area	C-7	*

Table 35. Refuels At Relationship

ATTRIBUTES	*FORMAT*	
Base	C-8	*
Aircraft Tail Number	I-3	*

Table 36. Based At Relationship

ATTRIBUTES	*FORMAT*	
Base	C-3	*
Refueling Area	C-7	*
Time Refueling to Base	I-3	

Table 37. Time Booms to Base Relationship

ATTRIBUTES	*FORMAT*	
Refueling Area	C-7	*
Target Region	C-1	*
Time Refueling to Tgt Rgn	I-3	

Table 38. Time Tgt to Booms Relationship

ATTRIBUTES	*FORMAT*	
Base	C-3	*
Target Region	C-1	*
Time Base to Tgt Rgn	I-3	

Table 39. Time Tgt to Base Relationship

ATTRIBUTES	*FORMAT*	
Target Region	C-1	*
Target Region 2	C-1	*
Time Tgt Rgn To Tgt Rgn	I-3	

Table 40. Time Tgt to Tgt Relationship

ATTRIBUTES	*FORMAT*	
Base	C-3	*
Flight Call Sign	C-4	*
Type CAP Aircraft Used	C-4	*

Table 41. CAP Takes Off From Relationship

ATTRIBUTES	*FORMAT*	
Target Number	I-3	*
F Strip Number	I-2	*
Position Nbr Within FStrip	I-1	

Table 42. Consists Of Relationship

ATTRIBUTES	*FORMAT*	
Track	C-1	*
Target Region	C-1	*
Order with Track	I-1	

Table 43. Covers Relationship

ATTRIBUTES	*FORMAT*	
Target Number	I-3	*
Counter Attack AC Location	I-3	
Number of Ctr Attk AC	I-2	

Table 44. Counter Attack Base Entity

ATTRIBUTES	*FORMAT*	
Base	C-3	*
Type of Aircraft	C-4	*
Nbr of A/C Deployed	I-2	

Table 45. Deployed by JPlan Relationship

ATTRIBUTES	*FORMAT*	
Target Region	C-1	*
Target Number	I-3	*
Order of Diverted Tgts	I-2	

Table 46. Has Diverted Tgts Relationship

ATTRIBUTES	*FORMAT*	
Type of Aircraft	C-4	*
Weapon Load Number	I-2	*

Table 47. Can Carry Load Far Relationship

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*
Track	C-1	*

Table 48. Flies Relationship

ATTRIBUTES	*FORMAT*	
F Strip Number	I-2	*
Flight Call Sign	C-4	*
Pass Nbr within Mission	I-1	

Table 49. FStrip Pass Relationship

ATTRIBUTES	*FORMAT*	
Flight Call	C-4	*
Pass Nbr within Mission	I-1	*

Table 50. FStrip Sub Mission Relationship

ATTRIBUTES	*FORMAT*	
Type of Aircraft	C-4	*
Nbr of Support A/C	I-1	*
Type of support A/C used	C-4	*
Prob of No Area Attrition	F-3	

Table 51. NAA Weak Entity

ATTRIBUTES	*FORMAT*	
Target Number	I-3	*
TAR Request Nbr	I-1	*

Table 52. Are Against Relationship

ATTRIBUTES	*FORMAT*	
Type of Weapon	C-3	*
Weapon Load Number	I-2	*
Weapon Quantity	I-2	

Table 53. Is Made Of Relationship

ATTRIBUTES	*FORMAT*	
Target Number	I-3	*
Interceptor AC Location	I-3	
Nbr of Interceptor AC	I-2	

Table 54. Interceptor Base Entity

ATTRIBUTES	*FORMAT*	
Number of AC Over Target	I-1	*
Pass Number	I-1	*
Time In Target Area	I-2	

Table 55. Time in Tgt Area Relationship

ATTRIBUTES	*FORMAT*	
Type of Aircraft	C-4	*
Weapon Load Number	I-2	*

Table 56. Can Carry Load Near Relationship

ATTRIBUTES	*FORMAT*	
Target Number	I-3	*

Table 57. Is of Tgt Type Entity

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*
Target Number	I-3	
Pass Number	I-1	
Type of Weapon	C-3	*
Weapon Quantity	I-2	

Table 58. Pass Relationship

ATTRIBUTES	*FORMAT*	
Type of Aircraft	C-4	*
Type of Target	C-1	*
Type of Weapon	C-3	*
Nbr of Bombs per Pass	I-2	*
Prob of Damage	F-3	

Table 59. Damage Prob Relationship

ATTRIBUTES	*FORMAT*	
Type of Weapon	C-3	*
Type of Aircraft	C-4	*
Wild Weasel Used	C-3	*
EWPOD USED	C-3	*
Prob of NAT No AAA	F-3	
Prob of NAT Light AAA	F-3	
Prob of NAT Medium AAA	F-3	
Prob of NAT Heavy AAA	F-3	
Prob of NAT SAM	F-3	

Table 60. NAT Probs Relationship

ATTRIBUTES	*FORMAT*	
Type of Aircraft	C-4	*
Attack Profile	C-7	*
Prob of NGE No Camouflage	F-3	
Prob of NGE Light Camouflg	F-3	
Prob of NGE Heavy Camouflg	F-3	

Table 61. No Gross Errors Probs Relationship

ATTRIBUTES	*FORMAT*	
Type of Weather	I-2	*
Type of Recce Target	C-7	*
Weathers Effects on Rec Tgt	F-3	

Table 62. Effect On Recce Tgt Types Relationship

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*
Refueling Area	C-7	*

Table 63. Rec Refuels At Relationship

ATTRIBUTES	*FORMAT*	
Base	C-3	*
Type of Aircraft	C-4	*
Nbr of Spare A/C Reserved	I-2	
Nbr of ADA A/C Reserved	I-2	
Nbr of CAS A/C Reserved	I-2	
Nbr of Active A/C Reserved	I-2	

Table 64. Reserved For Relationship

ATTRIBUTES	*FORMAT*	
Game Cycle	I-1	*
TAR Load Number	I-2	*
Type of Aircraft	C-4	*
Type of Weapon	C-3	
Weapon Quantity	I-2	

Table 65. Is Made Up Of Relationship

ATTRIBUTES	*FORMAT*	
Target Number of Runway	I-3	*

Table 66. Runway Entity

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*
Pass Nbr within Mission	I-1	*

Table 67. Recce Sub Mission Relationship

ATTRIBUTES	*FORMAT*	
Flight Call Sign	C-4	*
Pass Nbr within Mission	I-1	*

Table 68. Target Sub Mission Relationship

ATTRIBUTES	*FORMAT*	
F Strip Number	I-2	*
Flight Call Sign	C-4	*
Pass Nbr within Mission	I-1	

Table 69. Target Pass Relationship

ATTRIBUTES	*FORMAT*	
F Strip Number	I-2	*
Flight Call Sign	C-4	*
Target Number	I-1	*

Table 70. Has Results Relationship

ATTRIBUTES	*FORMAT*	
Type of Aircraft	C-4	*
Game Day	I-1	*
Nbr of A/C In Commission	I-2	
Nbr of A/C in Maintenance	I-2	
Nbr of A/C with Battle Dam	I-2	
Nbr of A/C Dead	I-2	

Table 71. Status Kept In Relationship

ATTRIBUTES	*FORMAT*	
Wild Weasel A/C Used	C-3	*
EWPOD Used	C-3	*

Table 72. Supprn Equipment Entity

ATTRIBUTES	*FORMAT*	
Base	C-3	*
Flight Call Sign	C-4	*

Table 73. Takes Off At Relationship

ATTRIBUTES	*FORMAT*	
Type of Aircraft	C-4	*
Aircraft Tail Number	I-3	*

Table 74. Is Off Type Relationship

ATTRIBUTES	*FORMAT*	
Base	C-3	*
Game Day	I-1	*
Start Time Base Closed	I-4	
End Time Base Closed	I-4	

Table 75. Weather Conditions Weak Entity

ATTRIBUTES	*FORMAT*	
Base	C-3	*
Flight Call Sign	C-4	*

Table 76. WW Takes Off From Relationship

ATTRIBUTES	*FORMAT*	
Aircraft Tail Number	I-3	*
Flight Call Sign	C-4	*

Table 77. WW Assigned To Relationship

ATTRIBUTES	*FORMAT*	
Type of Weather	I-2	*
Type of Weapon	C-3	*
Weathers Effects on Wpn Prob	F-3	

Table 78. Effect On Weapons Relationship

Appendix B. *SQL Command Files*

The following figures contain the SQL commands used to transfer data from the JPLAN database to the FAST STICK database.

```
drop base_ac;
\g
```

Figure 13. DROP.BAC Command File

```
create table base_ac(ab_pod,
                     tac_ac_name,
                     num_tac_ac)
as select s.ab_pod, t.tac_ac_name, t.num_tac_ac
   from air_tpfdd s, tac_ac_utc t
   where s.utc \( = \) t.utc
\g
```

Figure 14. CREATE.BAC Command File

```
update base_ac
set ab_pod = 'PRI'
    where ab_pod = 'PRM'
\g
update base_ac
set tac_ac_name = '111A'
    where tac_ac_name = 'EF111A'
\g
```

Figure 15. UPDATE.BAC Command File

```
delete from deployed
\g
```

Figure 16. DELETE.DPL Command File

```
insert into deployed(base,type_aircrft,nbr_ac_deply)
select ab_pod,tac_ac_name,num_tac_ac
    from base_ac
    where tac_ac_name = 'F16A' or tac_ac_name = 'F4G'
     or tac_ac_name = 'F4E' or tac_ac_name = '111A'
     or tac_ac_name = 'RF4C'
\g
```

Figure 17. INSERT.BAC Command File

Appendix C. *Menu for Conversion Program*

```
                    FAST STICK FILE LOADING PROGRAM

          This program loads the FAST STICK files with data from
                          the INGRES database

                      Enter key for file to be loaded:

F1 : AIRBASE.DAT                        F2 : CONSTANT.DAT

F3 : ENEMY.DAT                          F4 : PROBS.DAT

F5 : RTARGET.DAT                        F6 : WEATHER.DAT

F7 : ATARGET.DAT                        F8 : EVENT.DAT

F9 : BATTLE.DAT                         F10: TAR.DAT

c  : CASEVENT.DAT                       a  : ALL FILES

                        x : EXIT PROGRAM

Enter :

Status :
```

Figure 18. Conversion Program Main Menu

Appendix D. *Scenario Generation Application Screens*

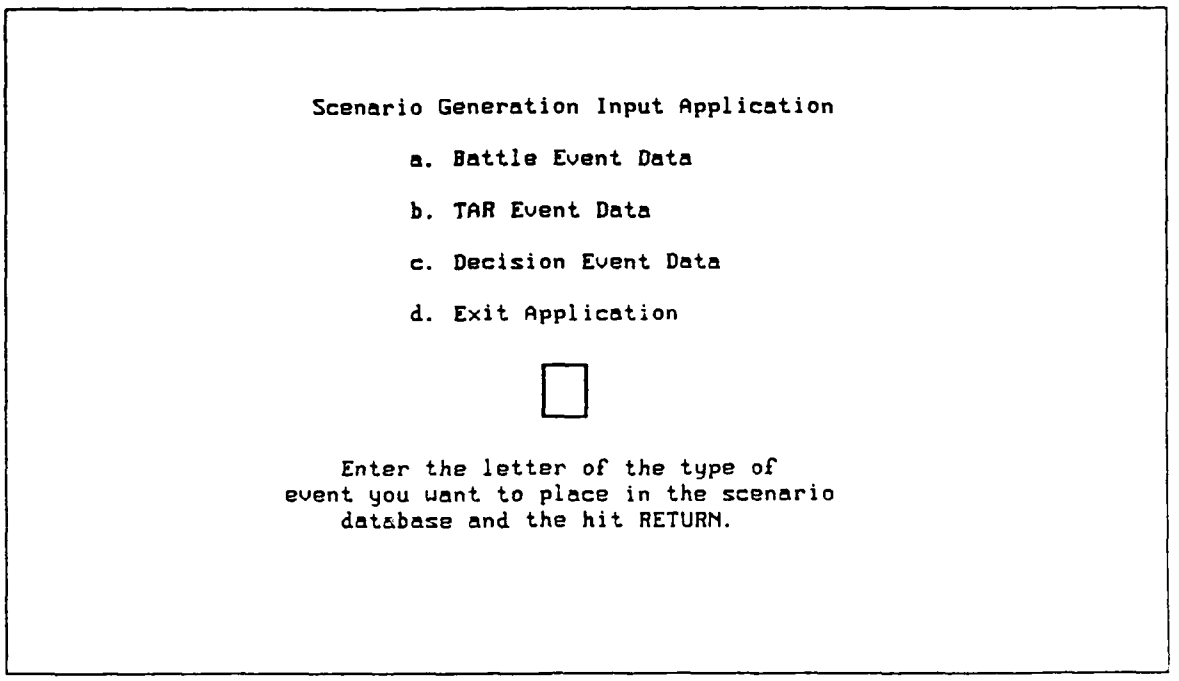

Figure 19. Scenario Generation Input Main Menu

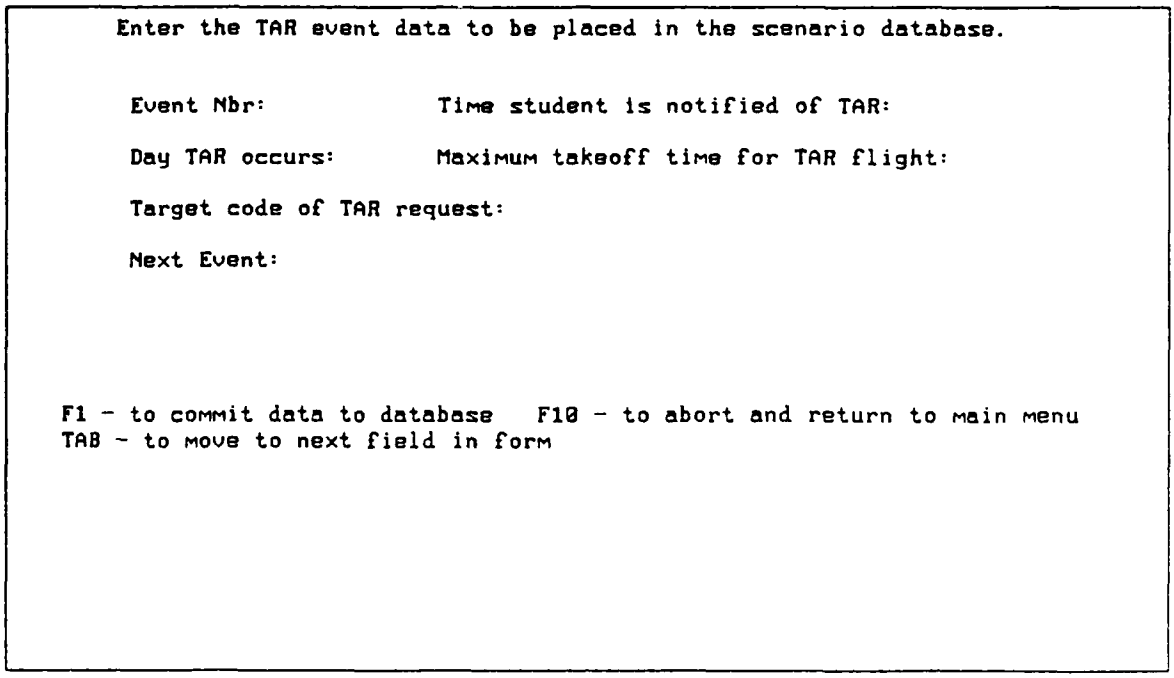

Figure 20. Tar Event Input Screen

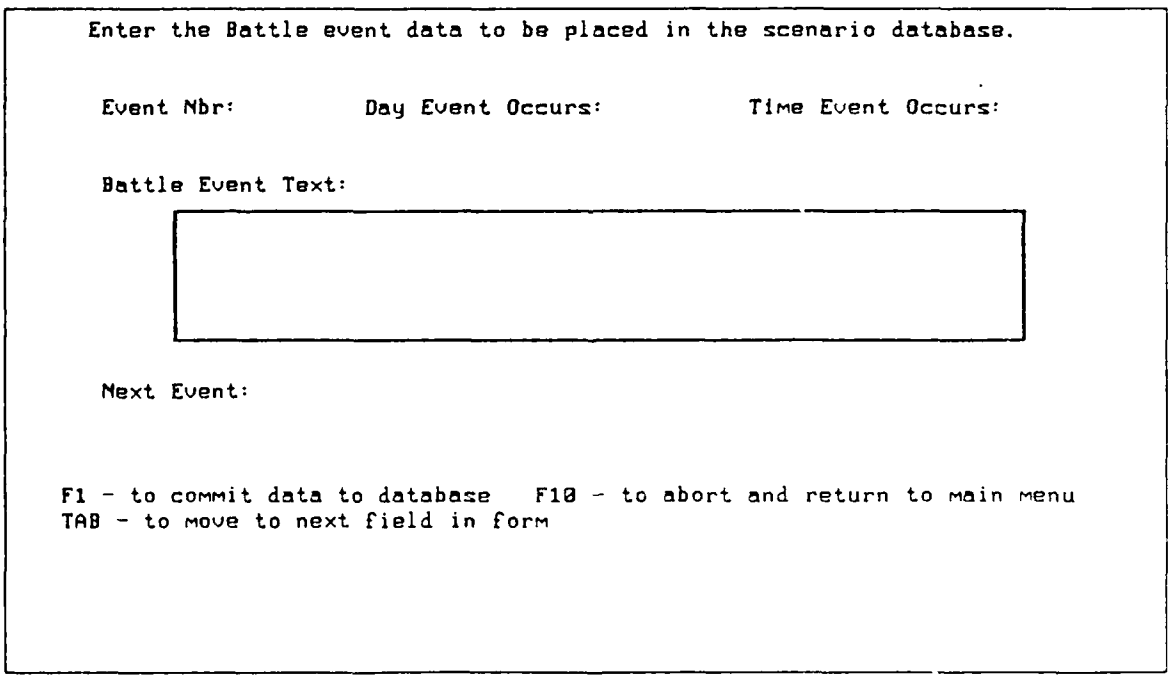

Figure 21. Battle Event Input Screen

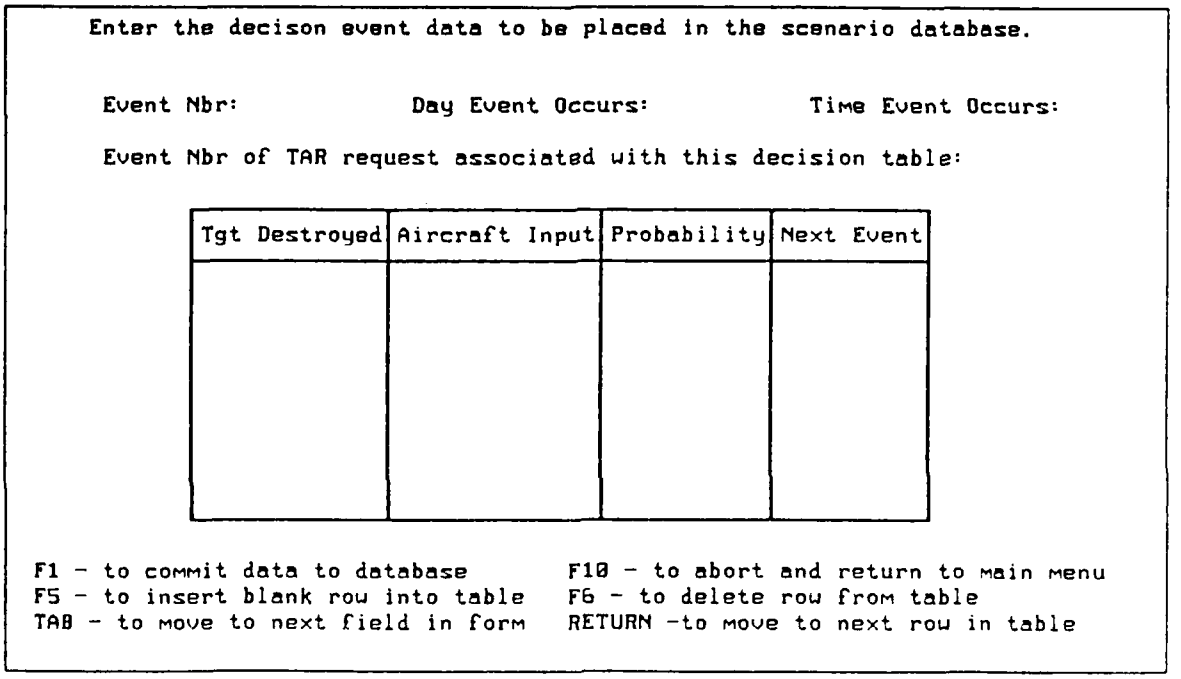

Figure 22. Decison Event Input Screen

Bibliography

1. *ER Designer.* Chen and Associates, Inc., Baton Rouge, La., June 1988.
2. *FAST STICK: A Tactical Air Forces Employment Feasibility Exercise.* Air Force Wargaming Center, Maxwell AFB, AL, 1987. Unpublished Manual.
3. *Joint Operation Planning and Execution.* Air Force Wargaming Center, Maxwell AFB, AL, 1987. Unpublished Manual.
4. *PC INGRES 4Gl Application Development Guide.* Relational Technology, Inc., Alameda, Ca., May 1987.
5. *PC INGRES Embeded Language Programming Guide.* Relational Technology, Inc., Alameda, Ca., May 1987.
6. *PC INGRES Reference Guide.* Relational Technology, Inc., Alameda, Ca., May 1987.
7. Captain Mark A. Roth, Assistant Professor of Computer Systems, Electrical and Computer Engineering Department, School of Engineering. Personal Interviews. Air Force Institute of Technology, Wright-Patterson AFB, OH, 1 June 1988 through 1 October 1988.
8. John A. Battilega and Judith K. Grange, editors. *The Military Applications of Modeling.* Air Force Institute of Technology Press, Wright-Patterson AFB, OH, 1984.
9. C. J. Date. *An Introduction to Database Systems.* Addison-Wesley, Reading, MA, 1986.
10. Lt. Col. Daniel B. Fox. *A Conceptual Design for a Model to Meet the War-Gaming Needs of the Major Commands of the United States Air Force.* Technical Report AU-ARI-84-8, Airpower Research Institute, Air University Press, Maxwell AFB, AL, July 1985.
11. J. K. Hartman. Lecture notes in aggregated combat modeling. 1985. Unpublished Notes.
12. J. K. Hartman. Lecture notes in high resolution combat modeling. 1985. Unpublished Notes.
13. D. R. Howe. *Database Analysis and Design.* Thomson Litho Ltd, Kilbride, Scotland, 1983.
14. Captain James Jansen. *Redesign of the Joint Planning Exercise.* Master's thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1987. AFIT/GCS/ENG/87-18.
15. Henry F. Korth and Abraham Silberschatz. *Database System Concepts.* MacGraw-Hill, New York, 1986.
16. A. Alan B. Pritsker. *Introduction to Simulation and SLAM II.* Halsted Press, New York, third edition, 1986.
17. Richard P. Smith. Computer assisted wargaming at the air university. In *Proceedings of the 1981 Winter Simulation Conference*, pages 679-684, IEEE Press, New York, 1981.

18. Dionysios C. Tsichritzis and Frederick H. Lochovsky, editors. *Data Models.* Prentice Hall,Inc, Englewood Cliffs, New Jersey, 1982.

19. S. B. Yao et al. An integrated approach to database design. In G. Goos and J. Hartmanis, editors, *Lecture Notes in Computer Science*, Springer-Verlag, New York, 1982. Volume 132.

Vita

Captain Swen Walker [redacted] He graduated from Westover Senior High School in 1979 and attended North Carolina State University. He graduated in 1983 with a Bachelor of Science in Computer Science. Upon graduation, he received a commission in the USAF through the ROTC program and began active duty in October of 1983 assigned to the National Computer Security Center. At the Center, he served as a Computer Security Research Officer, responsible for conducting exploratory research and development to determine methods and techniques to secure database management systems. In May of 1987, he was assigned to the Air Force Institute of Technology.

[redacted]

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/88D-22	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Wargaming Center 8b. OFFICE SYMBOL (If applicable) AUCADRE/WG	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Maxwell AFB, AL 36112-5532	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

DATABASE DESIGN AND LAND BATTLE INTERFACE FOR THE FAST STICK EXERCISE

12. PERSONAL AUTHOR(S)

Swen A. Walker, Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS thesis	FROM _____ TO _____	1988 December	95

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Database Design Land Battle Modeling
12	05		
12	04		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Advisor: Mark Roth, Captain, USAF
Assistant Professor of Electrical Engineering and Computer Science

Approved for public release IAW AFR 190-1
12 Jan 1989

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Mark Roth, Captain, USAF	(513) 255-3576	AFIT/ENG

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Abstract:

FAST STICK is an interactive, conventional, tactical air employment war simulation. The model is a teaching vehicle which allows users to apply basic, tactical employment concepts of air superiority, interdiction, close air support, and reconnaissance. The primary user of this model is the Air Command and Staff College in their theater warefare curriculum. The current version of FAST STICK was converted from Honeywell mainframe Fortran to IBM compatiable, microcomputer PASCAL by programmer/analysts from the Air Force Wargaming Center in 1987. Although major improvements were made to the program, the exercise still had shortcomings and limitations as a joint exercise. Specifically, limitations were identified in the following areas: a) the simulation exercises' use of flat files to store data; b) the lack of an automated data interface between FAST STICK and its deployment planning counterpart, JPLAN; c) the lack of land simulation play in this joint exercise.

The goal of this thesis effort was to enhance the FAST STICK exercise by addressing these limitations. This was accomplished by incorporating a relational database management system (INGRES) into the exercise to allow game controllers to better manage the FAST STICK data. Using the system level facilities and the power of the SQL language found in the INGRES database management system, an automated interface was constructed between the new FAST STICK database and the existing JPLAN database. Finally, an extension to the FAST STICK simulation was developed that permitted game controllers to incorporate different land battle scenarios into the exercise.

END